Aalto University
School of Science
Master's Programme in ICT Innovation

Design Systems for improved development efficiency in Software Startups

Neža Đukić

Supervisor: Mika P. Nieminen, D.Sc. (Tech.)
Advisor: David Bauman, M.Sc. (Tech.)

Master's Thesis
Espoo, June 26 2020

AALTO UNIVERSITY
School of Science
Master's Programme in ICT Innovation

ABSTRACT OF THE
MASTER´S THESIS

Author: **Neža Đukić**

Title of the thesis: **Design Systems for improved development efficiency in Software Startups**

Number of pages: **94 + 4**

Date: **17.6.2020**

Major: **Human Computer Interaction and Design**

Supervisor: **Mika P. Nieminen**

Thesis advisor: **David Bauman**

The Design Systems concept originates from the need of IT industry to make design reusable, consistent and easier to maintain at a scale. In practice it consists of a set of deliverables, that continuously evolve with the product (e.g. style guides). Recently it has been gaining traction within large companies because it can improve the output of the software teams. However, there is very little research on the suitability of Design Systems for software startups.

This study aims to determine whether software startups should use Design Systems, what benefits they could have in this context and to provide guidelines for startups that attempt to build one. To better understand how software startups work, we also look into current practices, potential challenges, and previous experience with Design Systems. In this context, software startups are defined as companies younger than 8 years, with no more than 50 employees and software as part of their product or service with a preference for Mobile and Web technologies.

To get a thorough understanding, we combined Design System implementation for a software startup with qualitative interviews. The participants of the interviews were employees in various roles at 5 software startups. The analysis of the results showed that software startups are aware of the concept and in some cases even working on implementation. However, Design Systems are not useful for all software startups alike. Therefore, it is recommended that the decision about their suitability is need driven and made for every startup individually and based on aspects like timing, team and project structure. Since many software startups under-prioritise design in resource allocation and decision making, the Design System can serve as a way to put more focus on the design without compromising the development or at least synchronise the team efforts. The greatest benefits of a Design System for software startups are aligned with those of large companies and include increased design consistency, development efficiency and a clearer work structure.

Keywords: **Design System, Software Startup, Design, Mobile Development, Web Development**

Publishing language: **English**

Acknowledgements

To be completely honest, it is probably impossible to mention everyone that had at least a little impact on this thesis coming alive. Nonetheless I'll do my best to outline the most important contributors. To start with, I literally could not have done it without David Bauman, Olof Lundström and the rest of the incredible team at L2GO, who encouraged my exploration of this fascinating subject and understood its potential. My honest thank you also goes to everyone who participated in any of the phases, patiently answered endless questions or completed the given tasks and with that provided the data that I used to form conclusions. Thank you Patricija Brecko and Nejka Progar for using your expertise to provide valuable feedback and help me polish the interview process.

I've also had a great emotional support team. A big thank you to my wonderful family for their tireless love and support from the moment I announced my urge to study abroad up until now, when it is all about to end. I also want to thank my awesome boyfriend Simen Fossnes who motivated and inspired me whenever I lost focus or felt like it would never end. Furthermore, I'm really grateful for all my amazing friends that endured my senseless complaining and believed in me anyway. You are purely awesome!

Last but not least, I'd like to thank my incredible supervisor Mika P. Nieminen who got through all the raw and ugly versions of this thesis, kept me accountable and endured my monstrous, lengthy sentences. I have no idea how you managed to still give me useful feedback, but I am glad you did.

Table of Contents

Chapter 1

Introduction

"All Design Systems start the same: as an attempt at making order out of chaos."
Vesselov & Davis (2019)

Although the concept originated in 1962 when designers began to realise the benefits of having shared underlying rules (Ackoff 1962), Design Systems only recently started gaining traction, mostly in the IT industry (Beck 2017). Suffice to say that design changed a lot since then, but people are still driven by the need for order and control (Vesselov & Davis 2019). With the rapid progression of technology the need for systematising design is higher than ever (Kholmatova 2017). Even without a mutually agreed definition, the goal of a Design System is clear: to make design reusable, consistent and easier to maintain on a larger scale (Suarez et al. 2017). In practice that makes it a set of deliverables, that continuously evolve with the product and are derived from existing practices (e.g. style guides) (Hacq 2018). If executed properly, Design Systems can improve the workflow and output of software teams, which may partially explain why many successful large companies such as Salesforce, IBM, Google and Airbnb are using them (Kholmatova 2017).

There can be many different benefits of a Design System, depending on the organisation's needs and how it is structured. Commonly mentioned benefits include better consistency across an array of products, stronger brand identity, improved development speed, easier collaboration and on-boarding of new members (Bostian 2019*a*, Pyrhönen 2019). However, it is important to only consider a Design System if it can solve actual problems of the organisation (e.g. accumulated technical or design debt) as it might not be the right solution for everyone (Farino 2017). While most agree that it is a good investment for large companies showcasing a certain degree of user experience maturity (Kholmatova 2017, Nielsen 2006, Curtis 2017*b*, Mauduit 2019, Farino

2017), there is very mixed opinions on the effect of the Design Systems for smaller teams and startups. According to Kholmatova (2017), everyone from small to medium sized production teams can benefit from this approach, while Saring (2019*b*) warns about the effort and cost involved in the process. Wallas (2019) claims that Design System is too slow for the fast changing ideation processes of the startup.

To quote Reis (2011), a startup is *"a human institution designed to create a new product or service under conditions of extreme uncertainty"*. The startup trend is mainly attributed to the dot-com bubble from the late 90s, when Internet based companies grew rapidly as the technology gained in value (Walsh 2009). Even though it was followed by the dot-com crash that only a few survived, the web continued to grow and made it easier for the companies to reach customers (Walsh 2009). Hence, 21st century was accompanied with numerous startups, most technology oriented, appearing all around the globe (Walsh 2009). Consequently, the term began to be used very loosely, so it is really hard to define and frame the startup concept. In essence, the startup is still a young (no more than 10 year old) company with little to no revenue that often relies on some form of founding for its survival (Nisen 2014). According to TechCrunch, a startup is also limited by the amount of employees (no more than a 100), revenue rate and worth (Wilhelm 2014). For the purpose of this thesis, I will further limit this criteria to less than 8 years old with no more than 50 employees to provide a clear separation and exclude any possible edge cases. I will also be focusing in particular on startups with software as part of their product or service with a preference for Mobile and Web technologies. From now on, the companies fulfilling all of the criteria will be referred to as "software startups" for simplicity.

A common issue of being a software startup is the lack of resources. This can make it hard to prioritise design over the development process, despite a widening understanding of the benefits and importance of design for product success (Moroni et al. 2015). Furthermore, software startups generally deal with different web and mobile products, so they have to handle multiple projects, platforms and technologies. That makes Design System an interesting approach to potentially improve the efficiency of the entire workflow. The aim of this study is to determine whether a Design System is a suitable option for startups and if and how it can help solve or at least reduce some of their problems. This is done both through research of the current situation and potential for improvements as well as working on an actual use case by building a Design System for a software startup[1] through an internship.

[1] L2GO

The startup mentioned fits the criteria with 8 employees and 1.5 years of age at the time of writing. Its current projects consist of two service-based mobile applications, which makes it a good research case, and the two founders are curious about new technologies, including Design Systems. Due to the small team size, the Design System is the sole responsibility of mine as the thesis author under close supervision by one of the founders. The team contributes by providing feedback during the evaluation and authorising some of the bigger decisions. The final goal is to ascertain a set of guidelines, useful tools and technologies specific to software startups to make the process easier for them. The following is the specified scope of the research to ensure clarity.

1.1 Scope of the research

This research revolves around the benefits of Design Systems for software startups. I acknowledge that Design Systems in general are a very broad subject. There is also very little previous research done specifically on their relation to software startups, which is further discussed in Chapter 2. Therefore, I aim to use this study as a starting point for this under-researched area. I see a lot of potential and hope to open it up for further research.

My first goal for this thesis is getting insight into the work of the software startups and challenges they are facing, with the aim to explore if Design Systems might be able to solve some of them. The focus is evaluating if this approach (as it has evolved up until now) is even suitable for software startups. Furthermore, I want to go beyond the theory into practice and test it out on an actual case with the goal of developing a usable Design System. Through undergoing the process myself, I want to establish a set of useful guidelines that can be used for implementation within the industry.

However, as this work is limited in both time and capacity, I assessed that performing a thorough evaluation of Design System performance in practice is not possible within the scope of this research. The main cause of this is the nature of the Design System itself as a long term solution (Rastelli 2019). Hence, it may take a while before the benefits start to show (Kholmatova 2017), so trying to validate it over a short time period could falsify the findings. Instead I choose to only validate the Design System implementation by testing its usability and based on the final decision of the team to include it into their work process or not. However, I do evaluate each of its elements throughout the development process to ensure that each part serves its role to the full

extent. I elaborate on this process in Chapter 3. In order to direct the focus of the study towards useful results, the following research questions are established.

1.2 Research questions

The first part of the research is about understanding the current software startup practices and problems and their views and knowledge on Design Systems as of now. This can enable informed decisions during later phases and ensure working towards actual relevant solutions and learning from previous experiences.

Research Question 1: What is the level of awareness about Design Systems among software startups?

The question was further fragmented for simplicity and better understanding of what is necessary to answer it in full:

Research Question 1a: What are the current practices for design and development among the software startups?
Research Question 1b: Which challenges are the software startups facing?
Research Question 1c: How familiar are software startup employees with the Design Systems concept?

It is important to know what software startup teams are experiencing nowadays and combine this with in-depth knowledge about what Design Systems represent and consist of. With that knowledge I can determine what benefits the Design System could have for designers and developers, even the rest of the software startup team. Those potential benefits are important for finding out if the Design System is useful for software startups and understanding which problems one should aim to solve with it. This can in turn provide a realistic assessment of whether it is applicable to a specific use case. After all, the needs may differ on a company basis, so even if there is merit in this practice, not everyone might find it usable.

Research Question 2: What are the benefits of a Design System for software startups?

As mentioned earlier, no two Design Systems are the same, meaning that it can be customised to some extent based on the company needs. The established benefits

from Research Question 2 are used in the subsequent section to come up with guidelines on the Design System process. Those guidelines aim to maximise the potential benefits while keeping the creation process in line with the resource limitations of software startups and propose different tools and technologies suggestions that could fit the purpose best. Working on an actual use case and going through the process allows for additional experience and a clear, up to date awareness of the possibilities out there and the potential pitfalls.

Research Question 3: How should a software startup build a Design System to maximise its potential?

Since the process of building a Design System is an elaborate task with multiple objectives and challenges, this question was also divided for clarity:

Research Question 3a: How should the Design System be structured and what elements should it include?
Research Question 3b: What would be the best process to construct the Design System?
Research Question 3c: Which tools and technologies seem to be most useful for a software startup Design System as of now?

Through these questions the study yields conclusions that can hopefully serve as an answer to the basic question: if the software startups should use Design Systems to improve the efficiency of their work. Although the measurement and evaluation of the long term efficiency of the Design System are not part of this thesis, it should help the software startups evaluate the suitability of a Design System for their case.

Chapter 2

Background

Within this chapter, resources on the topic are studied with a purpose of getting an in-depth insight into how the design process evolved to the point of Design Systems, defining what a Design System is and explaining the concept in detail to expose multiple perspectives. Furthermore, I aim to establish how it should look like in practice and wherein lie both its benefits and drawbacks. This background study is done with a context of software startups in mind, so existing resources on Design Systems for startups are also taken into account.

It is important to note that as a fairly new approach (Beck 2017), Design Systems lack in academic literature, even looking for citations of the basic written literature on the subject (e.g. Kholmatova (2017), Curtis (2010), Vesselov & Davis (2019)) yields very few useful resources and mostly thesis works. Same goes for looking at the proceedings of the CHI conference for the last five years (from 2015 to 2019). Design Systems are mentioned in one study in 2016 proceedings (Saakes et al. 2016) and even there the phrase is used with another meaning. Even more so, articles found elsewhere are mostly outdated and treat the concept as purely design related, neglecting its close relation to the development process that took place over time.

To overcome this barrier, the study is expanded to include a variety of more populist materials such as videos and web articles from the industry professionals. All of the listed authors work in the field and were either early adopters of this approach or part of its initial development and growth. For all such references, I rely on the fact that they are previously referenced by similar research works as this one or they are written by the authors whose names continuously appear as a reference for the Design Systems resources.

Additionally, most of the search for the appropriate literature is done through a sim-

ple mapping study, which is a process of *"classifying primary research papers in that specific domain"* (Kitchenham et al. 2011). This is a good method for getting a literature overview of a broad topic, compared to the Systematic Literature Review or SLR (Kitchenham et al. 2011). The findings are presented in the following sub-chapters.

2.1 The origin of Design Systems

Design has been around for ages but its use, understanding and interpretation varied over time (Gibbons 2016), specifically, there has been a back and forth movement between the appearance and the functionality (Vesselov & Davis 2019). One is all about the visual aesthetics and the other focuses on the usability and simplicity (Vesselov & Davis 2019). Design Systems first appeared as an emerging need to pay attention to the functionality, not just the visual appearance and at the same time having a set of underlying rules that would give design structure (Ackoff 1962, Vesselov & Davis 2019). This idea has in turn fueled the companies to document their visual identities as brand manuals for the first time, leading by NASA's graphic manual in 1975 (Pyrhönen 2019).

Later on, design continued to evolve together with the technology, which led us to the web design we know today that originated from the rise of the World Wide Web by Tim Berners-Lee in 1989 (Frost 2016). The reason for this evolution is mainly attributed to the rise of personal computers in the 90s, when more people got access to the internet and started using it as a media form, which resulted in the demands for better styling options (Vesselov & Davis 2019). Those were given first in 1996 as CSS was announced (Vesselov & Davis 2019) and then elevated with the first draft of Web Content Accessibility Guidelines (WCAG) in 1998 (Pyrhönen 2019). Finally, with the rise of software libraries, such as YUI and jQueryUI in 2000s, design was extended to Javascript which enabled the creation of components, widgets and patterns (Frost 2016).

Through all of the above, design began to be systematised beyond the style guides or brand manuals known before and moved more towards programmable code. Furthermore, the metaphor of having web as a series of pages was broken down into components for the first time (Frost 2016). The approach of modularity was not an unknown concept, but used in web development it has the potential to utilise the interactive nature of the web instead of thinking about it only in terms of pages (Curtis 2010). It became clear that there was a lot of repetition of components across the programs the developers were building and that reusing them across products can shorten develop-

ment time and lower costs (Shirali-Shahreza & Shirali-Shahreza 2010) while producing high quality results (Sharma & Shirisha 2010).

At the same time, developers were trying to find more efficient ways to build products and agile, iterative methods began to replace the traditional waterfall development, which translated into design as well (Vesselov & Davis 2019). This became even more necessary as mobile platforms emerged in early 2010s (Beck 2017) because they brought on new requirements for responsive Design to adapt to multiple screen sizes and development for multiple platforms. Brad Frost (2016) wrote about a conceptual component framework he called Atomic Design in response, urging people to adopt component oriented development. He explained how to build, group and organise components to best fuel the large ecosystems (Frost 2016).

To make the development easier, different frameworks like Angular and React also emerged (Connolly n.d.), but it still became increasingly harder for the design to keep up with the development and retain cohesiveness across products, platforms and screens. To counter this issue Google published Material Design in 2014 (Pyrhönen 2019), which is considered a benchmark for Design Systems since they offered a joint repository resources and guidelines for user interfaces across a range of platforms (Connolly n.d.). Afterwards, the trend was soon adopted by other companies like IBM and Airbnb which brought us to the current stage, where Design Systems are established in the industry and slowly becoming a field of experimentation and research, led by people like Alla Kholmatova (2017) and Nathan Curtis (2010).

2.2 Defining a Design System

As previously mentioned, there is no widely accepted definition for the concept of a Design System, therefore I establish one to abide by in this research. We can approach defining a Design System as an abstract concept or from a more structural point of view by describing its sub parts. Following are the definitions from various resources that I use to establish a common ground for this research:

According to MacDonald (2019), *"a design system is a single source of truth for shared parts and processes to build consistent products that is tailored to organizational needs and reflects the culture, team values, and visual language of an organization."*

Pyrhönen (2019) lists it as *"a living system that helps organisations deliver consistent, on-brand experiences at scale and over time."*

It is a documented approach to systematic design that lets teams move faster by re-

ducing the layers of translation between design and implementation (Vesselov & Davis 2019).

The definition of Hacq (2018): *"Design System is a single source of truth which groups all the elements that will allow the teams to design, realise and develop a product. It constantly evolves with a product, tools and technology and its fundamental purpose is to facilitate team work."*

"Design Systems provide a centralized, convenient and evolving map of a brand's known product territories with directional pointers to help you explore new regions.", a quote by Chris Messina, former Developer experience Lead at Uber (Fanguy 2019).

Definition from the MarvelApp e-book given by Punchev & Williams (2019) describes them as a powerful approach to help scale design-led development of products and services.

Nathan Curtis (Farino 2017) views it as a living, funded product with a road map and a backlog, serving an ecosystem.

Toman (2017) describes a Design System as connecting a style guide and component library combined with the principles on how they should work together.

A very detailed definition by Vesselov & Davis (2019) states it is *"a series of documented elements, components, and regions that include both design and front-end guidelines. The documentation contains live code examples and the design system also includes underlying design principles, rules, and guidelines that help a team build one or multiple products."*

It contains all of the design, code, and content resources (MacDonald 2019). A set of connected patterns and shared practices for a digital product. (Kholmatova 2017)

According to Pyrhönen (2019), it can be anything from principles and goals, brand identity, functional patterns (design and code), guidelines, tools, examples and best practices.

"A Design System offers a library of visual styles and components, documented and released as reusable code for developers and tools for designers. May also offer guidance on layout, accessibility, branding, patterns, etc." (Curtis 2017*a*)

Stated by Punchev & Williams (2019), it can also be a set of reusable standards and components which reinforce a brand's identity.

Finally, a very relevant argument is that before deciding on what a Design System is for your organisation, you should ask *"Who is going to use it and how?"* (Hacq 2018).

It is safe to assume that every Design System will be a bit different and unique in structure based on the specific needs. However, there are common aspects in the

way people define it despite approaching it from different angles. I use the common denominators to establish the definition used in this research:

> A Design System is a single source of truth for designers and developers that evolves together with the product(s) over time, facilitates team work by reducing the layers of translation between design and development and reflects the organisation and its culture. It should include both the more abstract styles that represent the brand identity, the organisations' design principles and guides on best practices along with a well documented concrete representation of designs in code as components and patterns as seen in Figure 2.1.

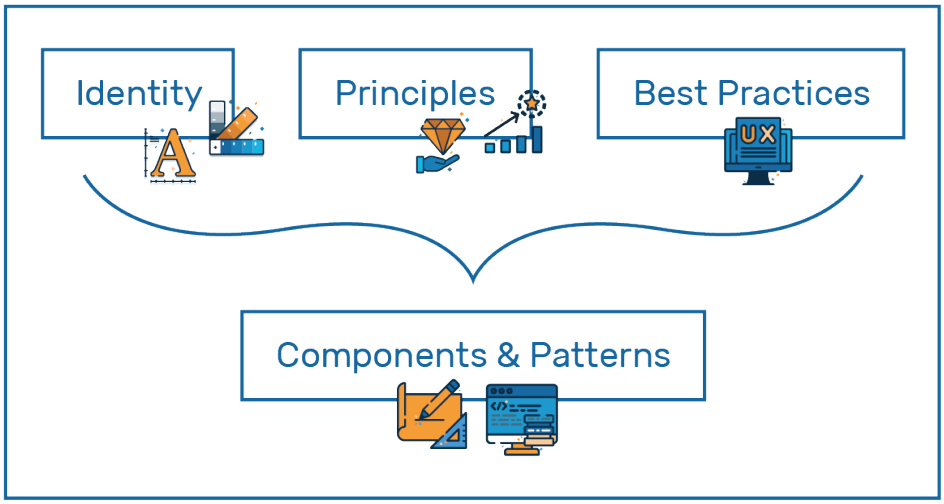

Figure 2.1: The structure and assets of a Design system.

In the next section we take a closer look at the common assets that are part of the Design System structure.

2.3 Design System structure

As previously established, Design System's structure differs based on the organisation's needs. This section explores some of the commonly used practices to achieve both the visual representation and the functional counterparts to easily build applications. In the following subsections I focus on each of the specific common elements of a Design System: brand identity and principles, style guide and design language, accessibility and usability guidelines, and component and pattern libraries.

2.3.1 Brand identity and Design Principles

Brand identity represents how a brand looks, feels and speaks to people and is mainly important to stand out from the competition and have an impact on the customers (Ghodeswar 2008). In a sense, it is a guarantee for the product or service based on the specific expectations of the public and can make the audience feel personally connected to the products or services that the brand offers (Knapp 1999). Therefore, it is important to have a clear vision of the brand that can then be used as the foundation for the entire Design System. It can be especially useful to determine the purpose, values and spirit of the brand (or even a specific product) and to have a core which informs all of our design and development decisions (Kholmatova 2017). Developing a brand identity can be approached in many ways, but commonly includes different design research techniques such as mood boards and style tiles (Ghodeswar 2008).

Mood boards are visual data tools used for inspiration, creating visual stories and organising the flow of thoughts and ideas. They can include anything, from imagery, colours, textures, text or even small objects that link to a theme, hence a mood board is nothing more than a space for the collected visuals (Cassidy 2011). As a branding tool, mood boards can be a good first visualisation of the vision that can then be interpreted into other segments (Kholmatova 2017).

Another approach for the brand identity development are the *style tiles*, which are focused on forming a common visual language between the designers and the stakeholders. According to the official style tiles website[1], style tiles are less vague than the mood boards and less precise than a mock-up of the product, so they should be used to establish a direct connection with the actual interface elements. They consist of fonts, colors and interface elements that are the essence of the brand and can easily be translated into a style guide. (Warren 2012)

On the other hand, the design principles are an extension of the brand identity, focusing on translating the purpose of the product into the design and development. For example, if the purpose is providing a more time efficient service, the necessary interactions should be kept to a minimum and very simplistic. Those principles can also be more product specific than the general brand identity, but they need to be a team commitment to achieve cohesive results. (Kholmatova 2017)

[1] http://styletil.es

2.3.2 Style guide and Design language

A style guide is a very common asset of the Design System. Vesselov & Davis (2019) define it as the *"static documentation that defines how the brand is stylistically applied to interface elements. It contains high-level details about color, typography, iconography, and more."* It is considered the most abstract part of the user interface and derived from the brand identity (Toman 2017). The main benefits of a style guide are consistency, everyone using the same language, having well-documented solutions for the common design problems (Hart 2000) and promoting good user interface practices (Park et al. 2011).

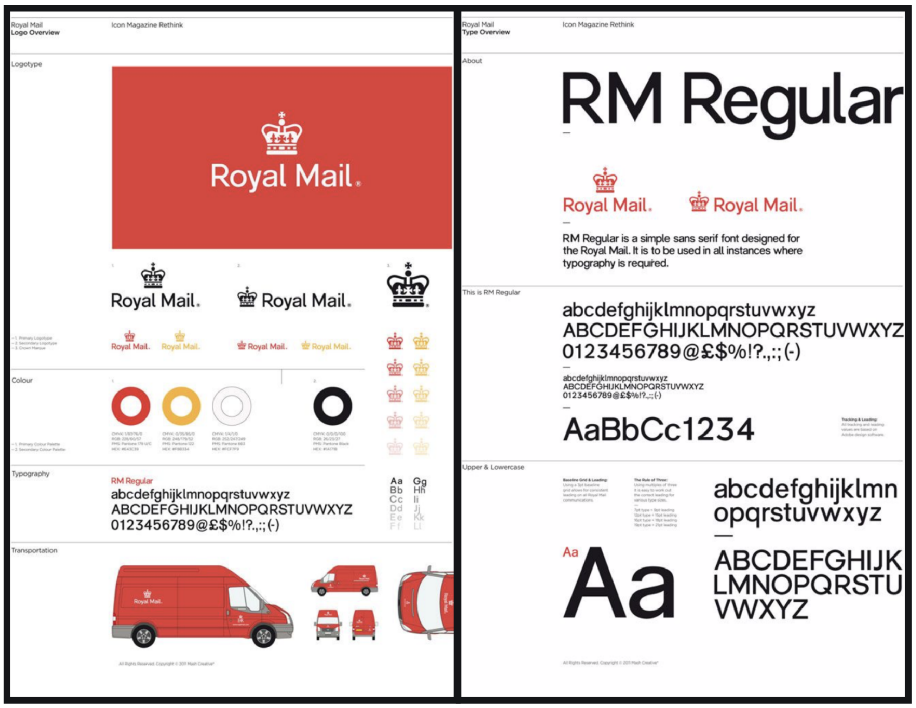

Figure 2.2: An example of the Royal Mail style guide (Mail 2011).

On the other hand, some argue that style guides are too general for usage and take too much time and effort to develop but end up not being that helpful. Hence, a lot of designers claim that they are not actually using it in practice (Park et al. 2011). An example of a style guide from Royal Mail is shown in Figure 2.2.

"Design language is a shared vocabulary for design." (Bostian 2019*b*)

Visual design language is recently becoming more popular since it is supposed to solve some of the issues of style guides. It can be a good asset for a Design System because it includes both the colors, typography, spacing and imagery, along with the

Figure 2.3: Design language from Ebury Chameleon Design System. (Hassan 2018)

visual interface elements (Fanguy 2019). The elements can vary from the more basic ones (e.g. a button) to complex components that are constructed from many elements. All of the visuals can be accompanied by guidelines on building products from the elements (Bostian 2019*b*). As is visible in Figure 2.3 of the Ebury design language, the design language is more complicated than a basic style guide.

Accessibility and usability guidelines can be part of the Design language or a separate asset of the Design System.

2.3.3 Accessibility and usability guidelines

According to Aveledo et al. (2010), *"usability is everything that plays a part in assuring that specific users can use a product to their satisfaction in order to effectively and efficiently achieve specific goals in a specific context of use."* With the Internet providing a global access to products and services, a variety of users might want to use them so it is very important to ensure they can. Therefore, arguments have been made to suggest that it should be part of the early development stages and strictly adhered to throughout the process (Aveledo et al. 2010). Gould & Lewis (1985) suggest the following principles of design for usability: early and continual focus on users, empirical

measurements of usage and iterative design. In practice this would result in a series of prototypes based on the characteristics of the determined persona(s) that are then tested for user feedback and improved in iterations until achieving satisfactory results.

While usability focuses more on making the products easy to use and highly functional so the users are able to achieve their goals, the accessibility guidelines specifically target users with disabilities and the special requirements they might have to be able to use something (Spillers n.d.). Quoted from the W3C (World Wide Web Consortium) (Shawn Lawton 2019): *"The Web is fundamentally designed to work for all people, whatever their hardware, software, language, location, or ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability."* To accomplish this, there is a set of WCAG (Web Content Accessibility guidelines) (Shawn Lawton 2018) set up to help organisations understand the accessibility for the web and the standards they should adhere to.

Nowadays, an increasing challenge is also accessibility in mobile development, seeing as mobile devices are increasingly used for a wide array of services. Darvishy (2014) establishes a list of accessibility features important for a mobile platform based on the different senses: screen reading, zooming capability, text magnification and bold text, colors and contrast options, speech rate options, etc. Those are a good start for consideration when thinking about accessibility in a Design System. The importance of accessibility is also recognised by EU with the Web Accessibility Directive in 2016 that enforces equal standards of accessibility for public sector websites and applications across the member states (Watson 2018). The deadline for website compliance is in September 2020 and for mobile applications in June 2021 (Watson 2018).

2.3.4 Component and pattern libraries

Both the component and pattern library represent the tangible part of the Design System and it is very important that they are well categorised and composed. In his book titled Atomic Design, Frost (2016) presents a mental model to help the authors think about the user interface composition. The latter should be seen both as a cohesive whole and a collection of parts. Those parts are divided into (Frost 2016):

- **Atoms**: the foundational building blocks of the UIs that cannot be broken down further like the basic HTML elements (e.g. buttons, inputs, etc.)
- **Molecules**: groups of atoms that give them a purpose, resulting in a simple, reusable component

- **Organisms**: relatively complex UI components that form sections of an interface (like headers or sidebars)

A component library is a set of common core elements (Vesselov & Davis 2019) that are responsive and independent but still mergeable (Toman 2017). The elements of the component library are also styled to reflect the specific brand they were created for (MacDonald 2019) and sometimes contain reusable code blocks which can vastly improve the development efficiency (Hacq 2018). To keep the Design System as a single source of truth, the code needs to be synchronised with any design modifications (Vesselov & Davis 2019). It is also very important to keep the documentation on the functionality of the components detailed, clear and up to date (Toman 2017). In essence, each component should be built for a domain specific reuse within a system (Curtis 2017*b*).

On the other hand, patterns represent a more abstract concept that emerged from a Pattern Language by Christopher Alexander, who proposed a content model for patterns (Alexander 1977). He argues that a pattern should describe a continuous problem and a core solution to it that can be reused for a million times (Curtis 2017*b*). A pattern library is, therefore, a set of building instructions for using components in a logical and consistent way (Hacq 2018) and applying design principles in action (Curtis 2010).

In practice, the two concepts can work hand in hand and are closely intertwined, so it might be hard to determine the difference at times. Whenever such an occasion occurs, a useful rule for differentiating is that components represent the UI, showcase how something works and define the given constraints. The patterns are about the user experience and describe how something should work by defining the conditions. (Curtis 2017*b*)

After exploring the content of a Design System, we look into the process of building one.

2.4 Building a Design System

"A Design System is a process and is therefore simultaneously always ready and never done." (UXPin 2017)

It can be tough to start building a Design System, no matter if it is from scratch or updating existing assets because it always requires a lot of changes (Vesselov & Davis 2019). Therefore, it is essential to get the support from the senior stakeholders

and sell the Design System to the organisation beforehand (Kholmatova 2017). Before starting the process, the organisation also needs to make a decision on the type of the Design System to build. Useful questions for discovering the best characteristics are: *"How many people will use it?", "What are their backgrounds?"* and *"For how many products will it be used? On which platforms?"* (Hacq 2018). Kholmatova (2017) based the shape of the Design System upon three attributes: *strictness of rules, modularity of parts* and *distribution of the organisation.* The attributes and their ranges are visualised in Figure 2.4 and further explained below.

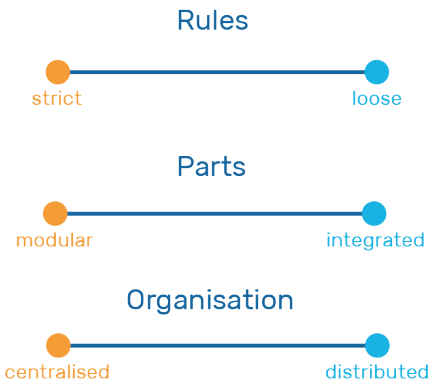

Figure 2.4: Design System characteristics.

Design System rules can either be strict or loose. Having strict rules means that the design and development are fully in sync, everything is documented in detail and there are strict processes in place. This helps in forming predictable, consistent outcomes, but it can also become rigid and outdated. On the other hand, loose rules are useful to have a more experimental process where the general brand feeling is more important than perfect visual consistency. The documentation involves sketches, not detailed specifications. This is good for discovering, however, it can become too messy and fragmented, especially if the team is not sufficiently aligned. (Pyrhönen 2019, Kholmatova 2017)

The next attribute to consider are modular or integrated parts. Building interchangeable or modular parts is good for agile development where different parts can be developed in parallel and makes the Design System easier to maintain and more cost effective in the long term. This approach can help teams scale and cater to multiple different user needs in an efficient manner. In an integrated Design System, everything is still made of parts, but they are not interchangeable, instead the content is made to fit the context, which is ideal for products with few repeating parts. It is also possible to

deliver it quickly since there is no need to consider reusability. (Kholmatova 2017)

Finally, every Design System needs to have a very clear organisation, either in a centralised or distributed form. Choosing a centralised form means that the management of the Design System is done by one group of people that both define the rules and patterns and have a veto over all of the System updates. The ownership ensures that the system will be maintained and evolved while this team also facilitates work for other teams. In a distributed system, the people who use it also contribute to the Design System and help maintain and evolve it. That makes the adoption easier and gives it more autonomy and resilience. After deciding on all of the characteristics, the implementation of the Design System can proceed as outlined in Figure 2.5. (Kholmatova 2017)

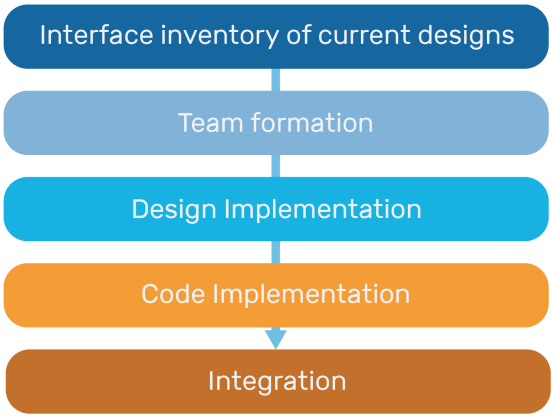

Figure 2.5: The process of building a Design System.

The first step is an interface inventory, that is conducting a visual audit of all current designs across products (Fanguy 2019) to understand the current state of the design and development ecosystem (UXPin 2017). Nathan Curtis (MacDonald 2019) suggests using his worksheet approach to decide which parts need solving and what people are needed for the work in collaboration. The team should agree on goals, objectives and the desired outcome to systematise the interface and establish shared processes (Kholmatova 2017). It can also be useful to screenshot and collect all of the different screens and pages to discover inconsistencies of the interface elements (MacDonald 2019). The inventory should also include the uses of colours, typography, spacing, etc (Punchev & Williams 2019).

Before proceeding, there must also be a team in place to build, integrate and maintain the Design System if choosing a centralised organisation. This team can then establish the rules and principles of the Design System (UXPin 2017), the baseline

components (usually the most used ones) to cover in the first version and the technology stack used to build it (Punchev & Williams 2019). From then on, the actual implementation starts usually by first focusing on design of the visual language (e.g. colours, typography, icons library, etc.) and then working on the component and pattern libraries (Fanguy 2019).

Usually, integrating the Design System takes an additional step because it can be a lengthy process of standardisation across existing products and integration into the processes (Punchev & Williams 2019). Besides building the Design System itself, its integration also requires a channel for knowledge sharing like instructions and tutorials or even workshops for new members (Kholmatova 2017). At the same time, it requires a good version control so the users can be aware of the changes and new features when it evolves over time. To maintain it and help it grow, it is also good to enable collaboration from an entire team through issue trackers or suggestion channels (MacDonald 2019).

2.5 The benefits and the drawbacks of Design Systems

"The reason for introducing a Design System is not so people can work less but so people can work better." (Rastelli 2019)

Design System as any other approach may offer different benefits and drawbacks that are studied in this section and are a good indicator of whether a Design System is a good opportunity for an organisation. However, the collection presented in this study is limited to the current findings about Design Systems. As those are mostly commonly implemented in larger companies, I examine the Design Systems in relation to software startups more closely in section 2.6. Furthermore, both benefits and drawbacks might differ based on the organisation and how they decide to implement it.

Before focusing on the specific benefits and drawbacks, it is also important to mention that the effects of such a system are very hard to measure in the first place. This is due to all of the different aspects of a Design System, the process it involves, which creates a delay in seeing results, and the fact that as a pioneer methodology there is no benchmark to compare against (Pyrhönen 2019). Furthermore, any organisation has a lot of different stakeholders with different values (Pyrhönen 2019) so any measurements based on the perception alone would result in biased, subjective outcomes (Rastelli 2019). So far, only a few possible approaches offering quantitative measurements have been suggested, such as using commit history and comparing the changed

lines of code and number of changed files between the component library repository and the main application repository (Rastelli 2019). Using those metrics, a study of the Cosmos Design System from Badoo showed reduction of work overall (supporting the benefits listed below) due to the reuse of the components (Rastelli 2019).

2.5.1 Benefits of Design Systems

The most common benefits are:

- Increased consistency
- Decreased maintenance
- Higher design efficiency and development speed
- Better collaboration
- Better products
- Built in accessibility

Having a Design System and standardised components, organisations can create more predictable UIs that are easier to use and feel familiar to their users (Suarez et al. 2017). That solves issues with brand consistency that especially larger organisations are facing (Pyrhönen 2019). This doesn't necessarily mean everything has to look the same but is focused on matching the solutions to similar problems and keeping them appropriate to the context (Beck 2017). Furthermore, with a Design System in place, the products remain consistent over time or with scaling (Bostian 2019*a*, Wallas 2019).

Design System as a single source of truth that translates into all of the products and services makes the maintenance much easier through trickle-down style updates (Bostian 2019*a*). Generally, it becomes hard to keep up and continuously update all of the products as the user interface standards evolve and aesthetic norms change (Beck 2017). Having non-reusable and inconsistent styles and conventions, which are the result of building for short term, the organisations acquire the so-called technical and design debt over time (Suarez et al. 2017). However, Design Systems offer a suitable solution. Furthermore, reusing means that there is also less components to maintain overall (Suarez et al. 2017).

With a Design System, there is no need to continue to design already existing components (Wallas 2019) and if it includes a component library, the developers don't have

to build them from scratch either (Bostian 2019*a*). This enables faster iteration when building new products and rapid prototyping using the premade ”lego blocks” (Suarez et al. 2017), which also reduces internal pain points and inefficiencies (e.g. the time (and cost) to the market) (Pyrhönen 2019). Since the assets work together from the get-go, people can spend more time creating and validating instead of documenting and detailing specifications (Pyrhönen 2019). All of the above results in a more efficient work process for both the developers and designers.

Having a single source of truth that everybody is using also fuels a more aligned organisation with less fragmentation (Pyrhönen 2019). Within large teams, a common problem is various opinions, cultures or even products, and a Design System can help facilitate communication in such cases (Beck 2017) by providing a shared decision-making criteria (Pyrhönen 2019). The core of it is defined through debates on what is considered good, so everyone can be involved and have a shared ownership (Pyrhönen 2019). At the same time, Design System also reduces the knowledge gap (Pyrhönen 2019), which is particularly helpful when on-boarding new team members who can then quickly catch on with the comprehensive documentation (Bostian 2019*a*). With a proper usage the teams are also kept in sync over time (Suarez et al. 2017).

Even products and applications can be improved using Design Systems. They keep the styling concise, which reduces potential styling conflicts in code and consequently decreases the overall page weight and loading time (Suarez et al. 2017). There is more time to focus on the individual components as well, so they can really be polished in terms of responsiveness for varying screen resolutions and viewports (Bostian 2019*a*). At the same time, components of a Design System can be highly optimised for users with disabilities, slow internet speeds or old devices (Suarez et al. 2017), especially as they are also based on the design principles of the organisation. As a result, every customer can use the products to achieve the same results (Bostian 2019*a*).

2.5.2 Drawbacks of Design Systems

On the other hand, Design Systems can have drawbacks as well:

- Time consumption
- Resources needed
- Need for support
- Creativity

Design System is a large, time consuming project with a lot of parts and it can take months or even over a year to get to a stable version (Bostian 2019*a*). Even more, it is never really a finished project as it requires constant maintenance and overview to produce the best outcomes (Bostian 2019*a*). And even as it is being used, it can take a while to get any positive effects (Wallas 2019). To get started, the Design System requires a lot of initial resources, which makes it a big, up-front commitment (Bostian 2019*a*). Within an organisation on a budget that might make it hard to sell to the stakeholders (Wallas 2019).

Furthermore, unless the entire team is on board and in agreement on having a Design System, it cannot work. Some of the problems it is supposed to solve are collaboration issues and it is very clear that there will still be inconsistencies if some part of the team continues the old practices (Bostian 2019*a*).

There is no consensus if Design Systems really kill the human creativity by setting very strict rules and practices in place. This is mainly a problem for UI Designers who might over time have less to contribute, but definitely not a problem for UX Designers who can ship prototypes much faster (Pyrhönen 2019).

To sum up, many of the above mentioned benefits and drawbacks are dependent on the organisation, the team and the culture. A Design System can definitely contribute to some positive changes in the workflow but, if executed improperly, might even make it worse.

2.6 Design Systems for software startups

There has not been a lot of research done about the usage of Design Systems for software startups. Therefore, most of the sources below are the opinions and experiences of people in the industry, who have worked on a Design System for a startup and not results from surveys or research. However, I argue that any data can be useful for the upcoming research, especially in a field as new as Design Systems where research materials are very limited so this section is still included in the study.

”Can you systematise something that is honestly still changing every day?”
(Johanessen 2018)

Opinions on Design Systems as a suitable method for startups are very mixed. The decision on their suitability should be based on the age of an organisation, team size, and volume and type of work (Vesselov & Davis 2019).

Regarding the organisation age, according to the research from Figma[2], a Design System usually comes after building the first product, as very strict guidelines are often unnecessary for a very young startup who continues to pivot to meet market demand (Vesselov & Davis 2019). However, some loose foundations can always be set even at that time to have a single source of truth going forward (Vesselov & Davis 2019). This coincides with Wallas (2019), who claims that it is a very slow method for the fast changing startup, so it might hinder creativity or even the entire company. On the other hand, waiting too long might mean that it will take exponentially more resources to undo the mistakes ingrained into the product (Johanessen 2018).

Team size matters because small teams usually cannot afford to work on such a large side project as their time is a limited resource (Vesselov & Davis 2019). Even if they end up building a Design System, the maintenance will cause continuous stray on the team (Wallas 2019). However, building a Design System before planning to extend and grow the team (anything beyond 5 designers and developers) is a good time to help get people on board and prepare for the growth with a comprehensive documentation on existing assets (Vesselov & Davis 2019). However, no matter the age and size of organisations, some work types are not as suitable for a Design System as the usual production teams, e.g. a consultancy or agency due to too varied, fast-paced and continually changing work (Vesselov & Davis 2019).

Figure 2.6: Traditional approach for building a Design System.

As most organisations have to prioritise building features over user experience in the beginning (Vesselov & Davis 2019), it might be best to make the Design System developer first by making the code the single source of truth (Saring 2019*b*). In general, it needs to include a strong brand and visual language and reusable code for fast development along with good foundations to scale (Saring 2019*b*). The Design System

[2] a UI/UX design tool company

should, therefore, not be fully fledged but rather formed organically to keep it light and easy to use (Wallas 2019). Contrary to the traditional approach of building a Design System in Figure 2.6, the approach for startups should be reversed so other projects don't have to be stopped in the mean time (Saring 2019*b*). The reverse approach (Figure 2.7) starts by designing and building components as a part of your application, auditing their design after and forming the Design System as a result (Saring 2019*b*). As a final step, the team is brought on board to collaborate and add to the components (Saring 2019*b*).

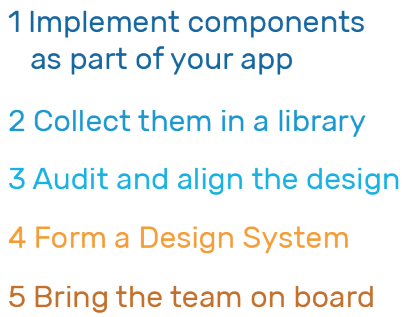

Figure 2.7: Reverse approach for building a Design System.

Another approach is designing in two threads: the main thread devoted to the features and products that the startup has to ship constantly, and the background thread for overall redesign of the product (Johanessen 2018). Time between the two can be divided based on the preference, but one quarter for the background thread works quite well to keep the team on track (Johanessen 2018). You should prioritise the components that are not too time consuming but commonly used, and measure success and course correct the whole time to ensure you end up with a good Design System (Johanessen 2018). Regarding the content, the team should avoid building "just in case components" that are not actually needed, but make the Design System heavier, and there should also not be too many variations for one component (Wallas 2019). Previously discussed pattern libraries are also too abstract and expensive in terms of effort for small teams and don't necessarily improve the cohesiveness for small teams (Curtis 2017*b*). The latter can instead benefit from only composing essential patterns unique to the user experience, if there are any, or sourcing patterns from the community, as good patterns don't just apply to one team (Curtis 2017*b*).

In conclusion, a Design System is not necessarily a good solution for small, young startups, but that doesn't oppose working on foundations for it in the mean time. If

startups decide for it nonetheless, they can find ways to include the workload into their basic work and make it lightweight and only as complex as is necessary. However, after the initial early stages, the Design System can be a great tool for the growth of the startup and drive a high quality production.

Chapter 3

Research Process and Methods

According to Dresch et al. (2015), research starts from realising that there is no structured knowledge on solving a specific question or problem. The objective for this research is closing the knowledge gap on the topic of Design Systems for software startups by providing new knowledge and making it applicable to both further research and usage within the industry. Hence, this thesis is empirical in nature and based on exploration with the goal to explore the main aspects and find novel hypotheses for future research (Jaeger & Halliday 1998). It was done along with the internship that served as a valuable source for the empirical data used to produce insights. I am aware of the importance of following procedures for the research work to guarantee that the results are reliable (Thornhill et al. 2009), hence I adhered to the concept of Design Science Research.

3.1 Design Science Research

Design Science was conceptualised from a need to introduce a science that is devoted to the creation of artifacts to attain specific goals for the purpose of areas, such as engineering, architecture and Information Systems (Simon 1996). The artifacts can be anything from systems, methods and models (Dresch et al. 2015). Contrary to the common misconceptions, its goal is not merely application, but also the generation of new knowledge about designing something properly (Dresch et al. 2015). Such research can extend the traditional ways of research to improving existing systems and the creation of new artifacts (Dresch et al. 2015). Although focused on problem solving, Design Science seeks adequate solutions suitable for the real world that can be further improved later, not necessarily the most optimal ones (Simon 1996).

According to Hevner et al. (2004), Design Science Research arises from a position that the *"truth and utility are inseparable. Truth informs design and utility informs theory."* This is the base of the conceptual framework for Information Systems in Figure 3.1 constructed by the same authors, which surrounds the two processes at the core of the research (*build and evaluate*) with the environment and knowledge base (Hevner et al. 2004, March & Smith 1995). *Environment* is a definition of the problem space that provides research with the business needs, and is comprised of people, organisations and technology (Simon 1996, Hevner et al. 2004). On the other hand, the *knowledge base* supplies the foundations and methodologies to use for the research (Hevner et al. 2004).

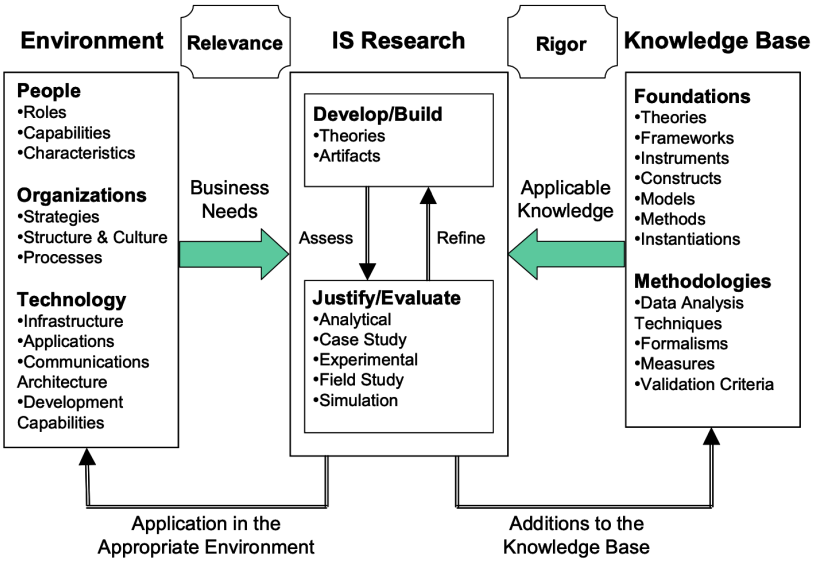

Figure 3.1: Conceptual framework for the Information Systems research (Hevner et al. 2004).

Addressing the right business needs enables relevance, while appropriately applying knowledge ensures rigor (Hevner et al. 2004). Both are equally important to produce results within the Design Science Research (Dresch et al. 2015) because *"a justified theory that is not useful to the environment contributes as little to the Information System as an artifact that solves a nonexistent problem"* (Hevner et al. 2004).

In this case, the environment consists of both the internship startup and the insights about software startups in general that I utilised to make the solution more applicable to the real world. I focused on the business needs of the software startups and used it to come up with my own version of a Design System. The research process was done as an iterative loop between development phases and evaluation of each step. Both were

informed by the knowledge base established in Chapter 2 and this Chapter. Finally, the goal of this research was to provide both a working Design System focused on software startups and add to the knowledge base, just as Hevner et al. (2004) suggests a proper Information Systems research should strive to.

As I aim for effective Design Science Research, I also followed each of the guidelines (Table 3.1) that were established by Hevner et al. (2004) to help researchers. There are 7 guidelines in total, the first one refers to producing an artifact that can be implemented in the given context, in this thesis that was a Design System for a software startup. The second one is about problem relevance as the researcher is supposed to solve previously unsolved and important problems. My objective was to explore an adaptation of the Design System approach for a software startup and evaluate its suitability. Guideline 3 poses importance on the constant evaluation of *"utility, quality and efficacy of the design artifact"* (Hevner et al. 2004). Since a Design System consists of multiple elements, I set a goal to proceed with an appropriate evaluation methods for each of them. The entire process of iterating towards the final solution is described in detail in section 3.2.

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 3.1: Design Science Research guidelines (Hevner et al. 2004).

It is also important to make a novel contribution (the fourth guideline) with the artifact itself, foundations or knowledge on design construction, and/or design evaluation knowledge. To keep in line with the scope I focused mainly on the artifact and producing foundations on the Design System construction for software startups in form of the guidelines for future attempts. I am aware of the importance of evaluation and the limitations that this omission creates for the validity of the produced Design System, but

instead I focused on thorough evaluation along the development process to argue that the solution still has merit. Guideline 5 focuses on the research rigor to conduct the research properly. To facilitate it I put large emphasis on following the right techniques and thinking about the artifact in regard to its generalisability, which I discuss later on.

To adhere to the sixth guideline, I approached the design as a search process and decomposed the problem into simpler sub-problems (Hevner et al. 2004). Not only did I develop the Design System in iteration as previously mentioned, but I also approached each separate element of it in the same way. To work efficiently I used the Synthesised Research approach (Figure 3.2) defined by Cole et al. (2005), because it was developed specifically for the Information Systems research. This method combines Design Science Research with Action Research, focusing on the development of the artifact within a real environment, where the researcher has to interact with other members of the organisation (Dresch et al. 2015). Action Research is a form of research, where the researcher is not only the observer but actively participates in the process and interacts with the researched object (Benbasat et al. 1987). In this context it was combined with the implementation of the artifact, the Design System and its elements.

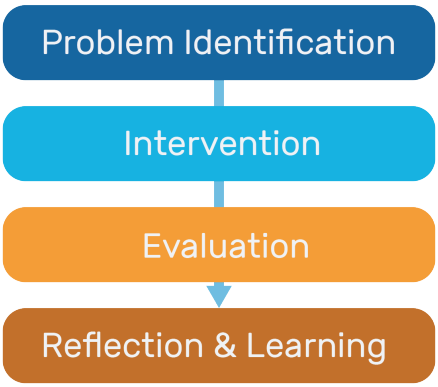

Figure 3.2: Synthesised Research Approach phases.

The process that Cole et al. (2005) advocates starts with the problem identification, which is supposed to help the researcher understand the problem and the personas involved, so I got acquainted with the needs for each specific Design System sub-part. Next step is intervention during which the artifact is created and implemented within the organisation (Cole et al. 2005). Hence, I proceeded to develop whichever element was the objective. I continuously added up towards the final objective of a functioning Design System. The development was also continuously evaluated as it was appropriate, based on the type of elements and through regular meetings with the software

startup. As mentioned, the specific methods are explained in section 3.2, however, the evaluation took into account the solution as well as the changes it introduced within the system and organisation (Cole et al. 2005). The last phase, which is about the reflection and learning (Cole et al. 2005), was executed through the analysis and reporting of the results within this thesis. This phase correlates with the last guideline of Hevner et al. (2004) to communicate the research.

3.2 Design System development phases and methods

To structure the process of building the Design System, I planned it ahead and also planned the methodology to assess the execution of each of the steps. I chose to try out the reverse approach suggested by Saring (2019*b*) to build the Design System along with the product development, in this case a mobile application. The predefined requirement for the Design System was compatibility with Ionic and Angular, which are the technologies the company is using for the development of the existing products. Since the development is most important to the startup right now, we also chose to keep it code first by giving the development side priority.

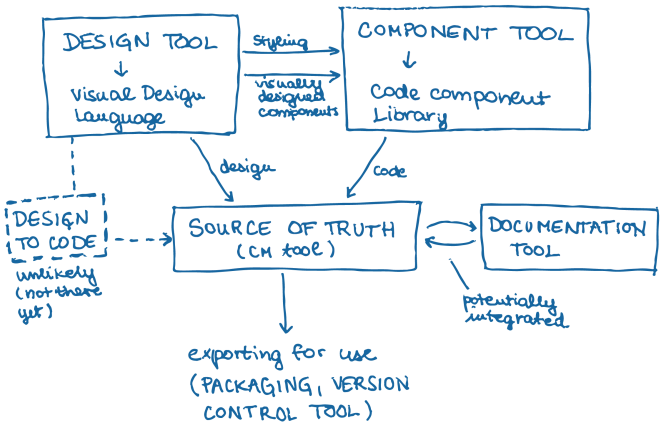

Figure 3.3: Sketch of the Design System structure.

As the company was in the process of recreating their brand identity, I started by working on a new brand to be used as the foundation for the Design System. After initial research, I decided to keep the Design System lightweight in the beginning. I primarily based it on a visual design language for the design tasks, integrating with a component library to support the developers and backed by a single source of truth and documentation. A sketch of the planned Design System structure is shown in

Figure 3.3. In order to build it, I followed the steps outlined in Figure 3.4, which are described in detail in sections 3.2.1 to 3.2.6. It is important to note that some of the steps overlapped at times, which made sense as the design and development work are usually intertwined and take place in parallel in iterative, collaborative work processes. Furthermore, such nature of work allowed me to make better decisions and produce higher quality results that fit well into the larger system.

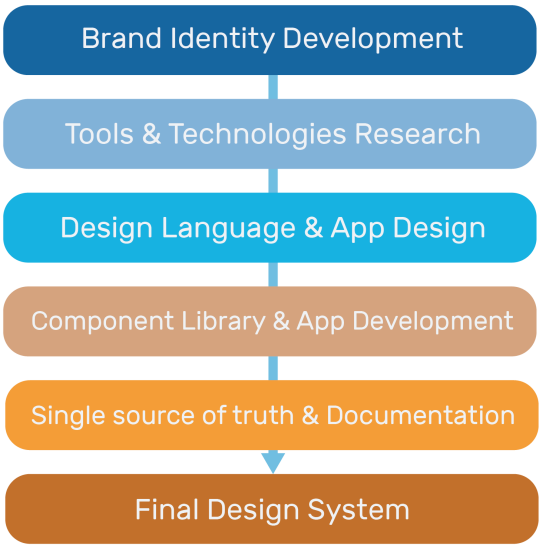

Figure 3.4: Design System Development process outline.

3.2.1 Brand identity

The brand identity was established based on the unstructured interviews with the two founders about how they want the brand to be portrayed. The conversation revolved around the message, purpose, values and spirit. Target audience was also discussed to make the brand and interface design suitable for them. Furthermore, to get a sense of the taste, a 20 second gut test method was used. This is a workshop designed to get an initial design direction (Allan 2015). It is conducted by giving the stakeholders a variety of designs that showcase the different styles and directions for the brand (Allan 2015). Each design is shown for 20 seconds and they are asked to rate it from 1 to 5 based on preference afterwards (Allan 2015). In the end, the scores are combined and the highest and lowest rated designs are further discussed to find out the preferences of the group (Allan 2015). In this case it was performed by the two co-founders and a third employee of the startup remotely on 15 randomly selected mobile application

designs from the Dribbble web page.

After the initial brand research, the brand identity development was done through iteration of mood boards (described in Chapter 2.3.1). All of the mood boards were first evaluated with the startup team members to see if they resonated with the vision and after a satisfactory version was established they were tested on a random selection of acquaintances of the team to see what associations arose. Hence, it could have been modified and polished to ensure a proper recognition. Finally it was used for the initial color scheme and typography that were further developed within the visual design language.

3.2.2 Research of available tools and technologies

To find the best tools and technologies, I took advantage of the analytical design evaluation methods as recommended by Hevner et al. (2004), mainly a combination of static analysis and architecture analysis. Static analysis is meant to *"examine the structure of the artifact for static qualities e.g. complexity"* while architecture analysis focuses on *"studying the fit of the artifact into the initial Information System architecture"* (Hevner et al. 2004). This comparison first took place to compare the design tool features for handling the Design System and then to find the best tool for its core (documentation and establishing a single source of truth).

The features that were compared are taking into account the nature of a software startup including a small team, uncertainty of the business and the idea, lack of funding and constant changes of the brand and product (McGowan 2018). Therefore, for design tools the features compared were pricing, ease of use, collaboration and how wholesome is the design tool. Ideally, it would cover all the needs for UI design, UX design and prototyping and have a good hand-off options. Additionally, I focused on the offer for the Design Systems, like component based design and version history. For the management and organisation of the Design System I compared organisation clarity, integration with the technology and designs, complexity of the tool usage and the approach to achieve a single source of truth.

In terms of choosing the technology, precisely the tool for the component library, there was no feature analysis done. Instead, I examined the tools used by famous existing Design Systems of large companies that also used component libraries and then took into account the architecture fit based on the requirements of the software startup. Lastly, for integration into the projects I went along with current most used practice that

works along all platforms for front end development - distributing the component library as a npm package that allows good version control of the components and is easily added to any project.

3.2.3 Visual design language and application design

The visual design language was based on the brand identity but extended beyond to also include the component designs used in the interfaces to help enforce reusability. The reverse approach by Saring (2019*b*), that I used to build the Design System, aims to develop components together with the design of an actual product to avoid creating redundant components. Hence, I went through this phase by designing an actual mobile application and then worked backwards to break down the screens into components. The objective of the initial application was to help manage the work of the cleaners performing the apartment cleaning, which is one of the services offered to the consumers. It was created in iteration and was based largely on the previously collected user feedback (not in the scope of this thesis). It also included some of my previous design work on the concept for the same startup which enabled knowledge and insights about the features and aspects that were important for the user experience.

The plan was to test the application with an actual future user in iteration through classic user testing sessions. The user was encouraged to think aloud while going through the prototype screens and give feedback both on unclear user flows and potential missing functionality. This plan was partially executed, however, as the work of the startup took a turn due to the exceptional circumstance of the Covid-19 epidemic in the spring of 2020, the priorities changed. Hence, the work of this thesis continued through rebuilding the existing customer oriented application instead.

The original application was built native for iOS and Android operating systems, but had to be rebuilt for the web to reach more customers, especially with a pandemic focused offers. Due to the time pressure and necessity, the decisions were mainly based on the previous, already tested version and familiar user flows. Even though there was no formal process of evaluation, the application was continuously tested internally in a testing environment by the startup employees who then gave feedback both on the design flaws and errors and issues with the code.

In parallel with the application design, the screens of both applications were used to define the components which then composed the visual design language. To break down screens into components, I relied on existing knowledge base to form an informed

argument for my decisions (Hevner et al. 2004). I followed the concept of Atomic Design, proposed by Frost (2016). He argues for a mental model in which we can consider our interfaces both *"as a cohesive whole and a collection of parts at the same time"* (Frost 2016).

Therefore, the components should be divided into 5 categories: atoms, the basic elements like buttons and inputs that we can't break down any further, and molecules or groups of atoms that work together (such as a search bar comprised of an input and a button) (Frost 2016). Next are the organisms that represent the more complex components or distinct parts of the interface (e.g. a footer) (Frost 2016). The last two categories are templates and pages, where the templates are page level component layouts (homepage template) and the pages specific instances of the templates with the real content (Frost 2016).

3.2.4 Component library and application development

The development of components was similar to and intertwined with their design. In this case each iteration was evaluated and given feedback on by the Lead Developer. Those evaluations took place in the weekly progress meetings and included discussions on good practices, code reviews and general feedback based on the application progress. As briefly mentioned in the previous section, when the application was at the far enough stage, it was also continuously tested by multiple employees for bugs, errors and browser compatibility. Based on the insights, the components were then refined and updated with more details. If necessary, the changes would go all the way back to the design. Throughout the building process of both components and the entire application screens, I also performed continuous functional testing, where I executed the interface to uncover potential defects (Hevner et al. 2004).

3.2.5 Single source of truth and Documentation

This step is very important for the usability of the finished Design System, but was only iterative to a point where it followed the development of the artifacts and was continuously updated as the Design System evolved. Therefore, it is not evaluated as a sub part, but in the end as part of the final Design System as described in the following section.

3.2.6 Final Design System

As previously mentioned, a proper evaluation of a Design System would require an analysis to happen over time as it mostly has benefits in the long term, which is out of scope for this thesis. However, a small experimental design evaluation was performed in the end, by studying it in a controlled environment for different qualities (Hevner et al. 2004). In this part, I focused on the specific important characteristics for a Design System that could be measured in the short term, such as providing a single source of truth, consistency and the ease of use (especially for people not acquainted with it beforehand). Those were tested through a traditional usability test (Nielsen 1993), where the user was a developer that was not part of the case startup or the Design System development process.

In the testing session the user performed a couple of tasks with different goals and was asked to think aloud during the experience. The tasks given in the instructions were to setup the component library in the project and then building two screens of an application using the single source of truth of the Design System (in this case the Storybook). The session took place online through a video call and the participant was asked to also share the screen throughout the test so the interactions were visible. This data was then used to reflect on the Design System along with the final decision of the company on its continued use.

3.3 Further qualitative assessment

In parallel with the classic Design System Research process (see in section 3.1), I also conducted interviews and qualitative analysis to aid the analysis and discussion. Contrary to the more factual nature of the quantitative research, qualitative approach enquires about meaning, perspective and experience of the participant (Hammarberg et al. 2016). Therefore, qualitative research is better suited for the exploration or when seeking a deeper understanding (Hammarberg et al. 2016), which made it a suitable approach in this case. This addition helped both to overcome the limited knowledge base and to better understand the environment I operated within and in turn enriched the Design Science Research. I argue that it also made the outcomes more generally applicable, as it offered insights into software startups in general, compared with only sourcing the information from one software startup.

3.3.1 Interviews

Qualitative interviews were chosen as the method to gather data because of their capability to explore a situation and or diagnose issues. This method takes advantage of the human conversation, where the participant answers a series of questions about a specific topic (Gubrium & Holstein 2001). An interview can be structured, e.g. follow a predefined script, unstructured when the researcher responds to the answers of the participants to explore the situation or semi-structured (Thornhill et al. 2009). The main benefits of this technique are the ability to rephrase the questions for better understanding and being able to observe other aspects like the attitude of the interviewee (Gubrium & Holstein 2001). However, interviews are subjected to the interpretation of both questions and answers, so this risk has to be mitigated in order to produce reliable results (Dresch et al. 2015).

This research utilised a semi-structured interview type by outlining a script to follow and allowing the researcher to ask specific sub-questions in case the need arose in the conversation. That way I could explore all the important topics, but also gained additional insights that would help me to better understand the situation. Using such technique also gave me an opportunity to elaborate on the questions if the answers were too scarce, so there was less of a chance of withheld information, which is a common interview drawback (Dresch et al. 2015).

Targeted participants for the interview were the employees of software startups that matched the constraints established in Chapter 1. To get an in-depth insight into the practices and challenges of the software startups, I sought different perspectives based on the role and background of the employees. As the Design Systems are most used and built by the developers and designers, I aimed to interview employees who work in those roles. However, I also sought the perspective of a decision maker of the company, hence, a person in a role of a founder and or a managing director to get their perspective.

For better comparison of the different perspectives and analysis of each startup as a whole, I aimed to interview at least one representative of each of the three roles within one startup. Nonetheless, smaller samples were still accepted to ensure that I would not pass on potentially valuable information and would be able to gather the sample size that could be representative when generalising the findings. When multiple participants from the same startups were interviewed, they were asked not to discuss it among themselves before all of the interviews were finished, and they were interviewed

in a random order based solely on their availability.

Every interview followed the same procedure and was executed through a video call. The participants were first acquainted with the researcher and the purpose and theme of the research and asked for the consensus on participation and data usage and specifically informed about the recording of the interview. The recording was done directly in the video call using the Zoom tool. The interview questions were prepared in advance and are attached in the Appendix A. The script consists of four parts: the initial questions about the participant and the startup were some more quantitative questions were included for proper placement of the startup and the employee. Some of the basic questions have been emitted if information on their background could have been gathered from their LinkedIn profiles. The second part is a section about their work (technologies, tools, practices, etc.) followed by the questions about the issues they encounter in their daily work and as a startup, and how they deal with them. The interview was concluded after the last section about Design Systems awareness and experience and lasted between 30 minutes to an hour.

Reaching out to startups from my network and EIT Digital Startup ecosystem, I managed to interview 13 people in total from 5 different startups, listed in the table 3.2. As it turned out, their roles were not as clearly divided as they might have been in the larger companies, so an employee commonly took on a combination of the roles. In that case they were asked the combination of questions suitable for their roles and analysed as representatives of both roles later on as well.

	Employee 1	**Employee 2**	**Employee 3**
Startup A	UX Lead	CEO, Co-Founder	Frontend Developer
Startup B	CEO, Co-Founder, Designer	CTO, Co-Founder	Lead Developer
Startup C	CTO, Co-Founder	Designer	
Startup D	Co-Founder, Developer	CEO	
Startup E	Director of Engineering	Lead Product Manager	iOS Developer

Table 3.2: Interviewed startup employees roles per startup.

3.3.2 Qualitative Data Analysis

"The process of analysis is one of transformation and interpretation." Seers (2012) citing Richards & Morse (2012)

To outline and summarise the findings of the interviews, I proceeded with a detailed qualitative data analysis of the interview transcripts using the Atlas.ti qualitative analysis tool that allows you to organise data and perform a systematic analysis, which is important to demonstrate rigor (Seers 2012). I started by transcribing the interviews using the Sonix tool for automatic audio to text transcription that were later audited to ensure their accuracy. Then I performed a combination of coding methods to sort the data for the later summary and interpretation of the findings (Newcomer et al. 2015). During the coding, parts of transcriptions were marked with codes, which are the words or phrases used to describe them (Newcomer et al. 2015). Using codes, wider categories were formed for emerging topics (Newcomer et al. 2015) and to summarise the data, those categories were used to form themes and more abstract concepts and relate them to the research objectives (Seers 2012).

Figure 3.5: Initial list of codes for qualitative analysis based on the research objectives.

The code framework can be derived a priori, from research questions or studied literature before analysing the data (Seers 2012) or inductively while going through the data (a concept of open coding) (Newcomer et al. 2015), but it is also common to combine the approaches (Seers 2012). The latter is what I used in this case, hence I started with a set of very general codes fitting the interview script topics and research objectives and then expanded and restructured it as I analysed the data. The initial list

of codes is listed in Figure 3.5 and was kept generic on purpose to make sure none of the main interests were overlooked in the material and on the other hand, not over-fit the data towards what I was interested in finding.

I focused on performing a thorough thematic analysis. According to Newcomer et al. (2015), thematic analysis is a process of *"identifying and eliciting manifest and latent meanings in the data."* The goal of this approach is finding the underlying similarities and differences to form meaning (Newcomer et al. 2015). For this thesis such analysis aided the understanding of how software startups work and what challenges they face to see if Design System would be a suitable approach. Hence, majority of the analysis was looking at the interview data and trying to understand it in relation to the Design Systems. To keep the emerging themes aligned with the research objectives, the codes were organised after each iteration of the coding and removed if they did not contribute to the overall understanding.

Chapter 4

Results

In this chapter I focus on the results of both building a functional Design System and the additional qualitative analysis of the interviews.

4.1 Design System Development

4.1.1 Brand identity

As mentioned in section 3.2.1, for the initial research unstructured interviews with the co-founders were conducted, where the conversation revolved around the message and values they wanted to communicate to the potential users. The interviews resulted in a clear purpose of the company to **make sustainability cosy**. The sustainability focus is about turning the linear consumption patterns into circular consumption without sacrificing the convenience of the users and helping them solve everyday problems they are facing. Furthermore, the aim is to bring **that nice hotel feeling into every aspect of the user's life**. The values and spirit that the brand is supposed to portray were defined in four parts.

First, it should feel **personal and trustworthy** in order to build a relationship with the user, especially since they need to feel comfortable inviting the startup employees into their homes. Some aspects of how that should be visible in the interface is to give user a consistent experience with clarity on the state of the system at all times and a good way to revert any mistakes. Second, it should be **cosy and heartwarming**, so the products and interactions should feel familiar and execution of the actions should be almost effortless to provide the convenience. Third, in regard to the looks, the brand should exude a **clean and fresh** feeling with highly usable interfaces, modern design

and no excessive, irrelevant elements. However, the cleanliness should not come off as clinical and cold. Finally, there should be a bit of **humour** in the brand as the experience should be enjoyable and lighthearted and not too serious or boring.

Another outcome of the interviews was the definition of the target audience as working adults with fast paced daily life, who seek convenience for services that would otherwise take up their free time. Based on the current customer base, there is no major prevalence of women or men, promoting a preference for a gender neutral brand that would continue to speak to all of them. Those findings were combined with the results of a 20 second gut test method (Allan 2015) to define the basic style preferences to take into account when working on the brand. Figure 4.1 represents the styles with the highest grades and Figure 4.2 the styles with the lowest grades.

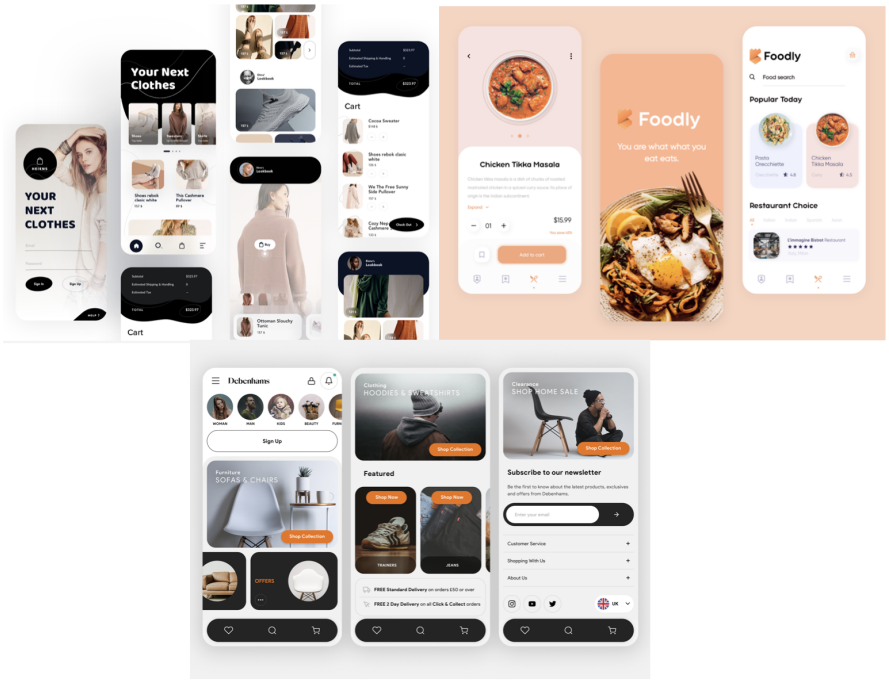

Figure 4.1: Highest graded interfaces in the 20 second gut test.

Some recognisable properties of the favourite interfaces are the use of neutral, soft or pastel colors with a grayscale base, rounded yet defined shapes and skeuomorphic[1] depictions of the products. On the other hand, the less preferred interfaces used stronger, saturated colours in combination with lots of white space. The conclusion was to find a colour scheme, inspired by the soft natural tones that would best fit the brand message. The process continued with an iteration of the brand mood board. The final

[1] skeuomorphism - interface items represent their real-world counterparts (Rouse 2013)

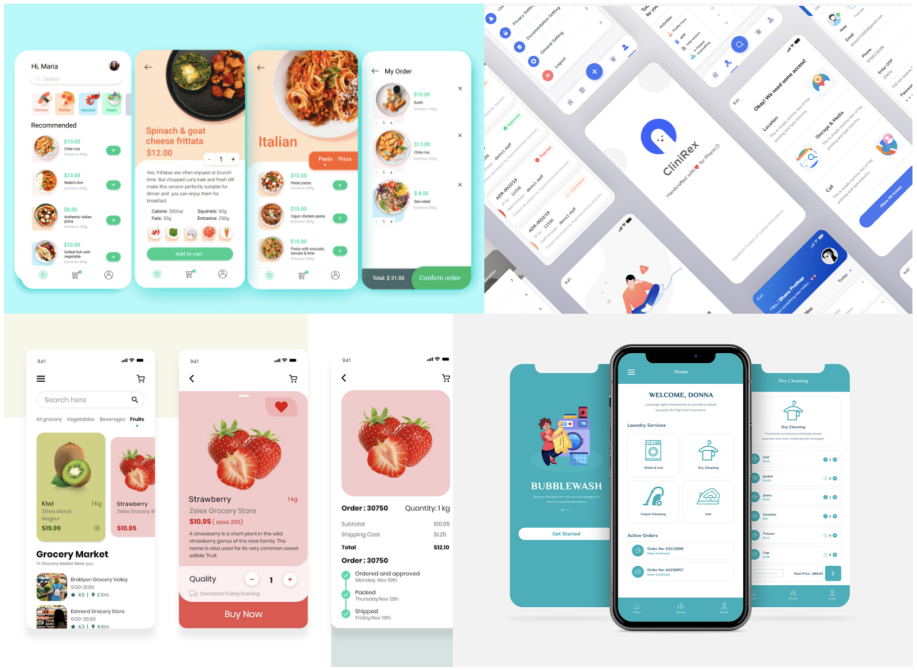

Figure 4.2: Lowest graded interfaces in the 20 second gut test.

version is shown in Figure 4.3. It includes a variety of images for the brand feeling, the initial colors that were used for a color scheme in the following phases and the typography. I chose blue color as a primary colour and added it to a variety of neutral colors. Blue color is associated with trust and security and was also a favourite colour of both male and female gender (Hallock 2003), which made it really suitable for the brand. For typography I chose a Google font Rubik, which was already used in the product logotype and is versatile enough to be used for all textual purposes.

Figure 4.3: Final version of the mood board.

4.1.2 Research of available tools and technologies

4.1.2.1 Design tools

For the design tool, four commonly used tools were compared: Figma, Adobe XD, InVision Studio and Sketch. The static analysis took into consideration the following features: price per year of usage, collaboration options, ease of use, wholesomeness of the tool (can you use it for UI design, UX and prototyping and or design hand-off) and finally, any specific optimisation for the Design Systems such as component based design and or version history. The features were assessed and rated from 1 to 3, where 3 is the highest score. The ratings are listed in the table 4.1 and summarised below.

	Figma	**Adobe XD**	**Sketch**	**InVision Studio**
Pricing	2	1	2	1
Collaboration	3	2	2	2
Ease of use	3	3	2	1
Wholesomeness	3	3	1	2
Design Systems options	3	1	3	1
TOTAL	14	10	10	7

Table 4.1: Feature comparison of design tools.

Figma has the best option of **pricing** for a small team as it is the only tool that actually enables team cooperation as part of the free version, if there are no more than two people working on the design and no more than three projects. For more people and or projects it still costs only a bit more than Sketch with $12/person per month. Sketch costs $9/person per month, so it makes a good pricing option as well, while Adobe XD costs $23/person per month so it is a bit more expensive. InVision studio is more expensive as well since it offers a static price for up to 5 team members for $99 per month and for further expansions there is an enterprise solution without a static price.

For **collaboration**, Figma offers a real time collaboration on team files, while the other tools offer collaborating and sharing styles and feedback across the designs, but not live co-design.

Regarding the **ease of use**, Adobe XD and Figma are very similar in the interface and experience and they are both very easy to use and have a steep learning curve. Sketch is a little bit more complex as big part of its functionality comes from different

plugins which the user also has to learn to use, but it offers a lot of useful resources and community support. A bit of a drawback is also that it is only available for Mac so it cannot be used on other computers. InVision Studio is more complex and is also a bit slower in performance, so it takes longer to load larger files. It also comes as a collection of tools: InVision web application for prototyping, InVision Studio for design and a specific Design System Management tool that is described in detail below.

For **wholesomeness** I took into account the 2019 Design tools survey (Palmer 2019), which describes what tools are most often used by designers for UI design, UX design and prototyping and hand-off. Figma was rated among the top three for all three and is also shown to grow in use the most from previous years, so it is gaining in popularity (Palmer 2019). It is a very wholesome tool and facilitates all the basic design requirements within a company. Adobe XD was rated a bit lower than Figma in the same categories, but is also improving and offers a lot of great features for more customised designs due to its integration with other Adobe tools. Furthermore, it also offers advanced animations for prototypes.

Sketch is really popular for both UI design and prototyping but only enables hand-off for designs through external plugins or directly in the program. The same survey shows that it is consequently not used for hand-off. It also offers only more basic animations by default. Finally, InVision offers all the functionality but through multiple tools, not InVision Studio alone. For prototyping and hand-off there is an InVision app and Inspector mode, so technically one still needs multiple tools for the three tasks. However, it does offer very advanced animation options and also the most variety in the hand-off.

Finally, I checked the features that could be **useful for a Design System**. At the time of writing none of the tools support any kind of direct translation of design to code, but they do offer some functionality that might enable better systematisation of design and better management and reusability across designs. That is mainly achieved by sharing assets, styles and or symbols across projects, which is a feature of Sketch, Figma and Adobe XD for teams. Figma also offers version history and a library of components that the designers can easily reuse and alter the child instances while remaining in sync with the original component. Such feature can go a long way in keeping the design consistency and Sketch has a similar one, but doesn't allow for changes to the child components.

InVision has a great Design System Manager tool, which is optimised for building a standardised system, manage design and share styles and components across teams. It also has the version history and even offers a way to generate style guides, however,

it is only integrated with Sketch and not yet with InVision Studio, which is currently in progress according to the company. That means that InVision Studio is not yet optimised for Design Systems and even to use this feature with Sketch it would still mean additional tools in the tool chain and consequently more work for the designer.

Taking into account the architectural fit, there is no big difference between the design tools because they offer very similar hand-off options, so there is no best tool depending on the technology used. Overall, the best choice was Figma because it offered all of the required features at a competitive price and made it possible to keep all of the design within one tool.

4.1.2.2 Component library tool

To choose the right tool for building the component library, I looked at examples of successful design systems from companies, mainly Google, IBM, Airbnb, Atlassian and Shopify. As it turns out, the most common framework used was React, which was used by Airbnb, Atlassian and Shopify. Those companies built their component libraries directly out of React components and made them available online. Contrarily, both Google's Material Design and IBM's Carbon offer support for mainstream frameworks, like React, Angular and Vue, and also classic web components that can be used anywhere within web development. That makes sense as those Design Systems are used by other companies as well for building products, especially Material Design that also sets recommendations and standards for the Android development.

Since the requirement for the component library was to be compatible with Angular and Ionic and there weren't many publicly available examples of Design Systems using Angular components, I also looked into the pre-built set of components provided within Ionic, which were built using Stencil.js. Stencil is a tool for building web components that can also be integrated with most mainstream frameworks for web development and is created by the same people who worked on Ionic, making it fitting to this use case and an equivalent alternative to React components. Therefore, Stencil was chosen for the component library, which made it possible to build some components on top of existing Ionic components to save time. It also makes it easy to switch frameworks later on if it would provide to be a good choice in the future.

4.1.2.3 Single source of truth and Documentation tool

There are a lot of tools available that provide suitable features, but they vary a lot in the functionality so they are hard to compare. Some of the tools also integrate and can be used simultaneously. I decided to compare the two tools that were mentioned repeatedly in sources like Saring (2019*a*) - Zeplin and Storybook.js. However, it is important to note that there are other options available.

Zeplin is used as a hand-off tool and is specifically focused on keeping a single source of truth for both designers and developers. It offers plugins for Figma, Adobe XD and Sketch to easily import designs and includes an option to connect the designs to components in the code using the feature called Connected Components and generate style guides for the company. It is very well organised and not too complex to use, although it requires quite a lot of work to maintain. Its main downside is that it is not so straightforward to integrate with the chosen development stack as there is no direct support for web components like there is for React.

Storybook.js is a very different tool with the original purpose to provide a sandbox for efficient component development. It facilitates the documentation of different use cases and even provides an interactive interface to render the components in different states. In practice, it looks like an interactive documentation page and there are many addons for additional information, such as *Notes* for readme files and *Design* to embed the design files directly from Adobe XD, Sketch or Figma. This is especially useful in this case because Figma provides a Live Embed function where the designs stay up to date with the current state in the Figma project. Storybook also supports all of the mainstream frameworks and is currently adding support specifically for web components though they can already be added using the basic html support.

Zeplin and Storybook can actually be used together to provide a more wholesome overview of the system. The Storybook is a bit more suitable for developers, while Zeplin is optimised for designers and hand-off. However, for simple, small systems using the two together requires a lot of work, especially maintenance. They both give the user ways to create a single source of truth and while Zeplin does it a bit better, Stroybook has better options for documentation. In this case, I decided to start out with a Storybook because it offered all of the needed functionality within one tool and it helped me make the Design System optimised for the development first. In practice, it was also easier to integrate with the web components. Additionally with this initial setup, Zeplin can still be added to the Design System as it grows if there is a need for

clearer management.

4.1.3 Visual design language and application Design

Figure 4.4: Initial (left) and final (right) color scheme.

The initial colors from the mood board were used to define a color scheme that was improved in iteration together with the interface design. The color scheme started out with a range of orange colours and brighter blues that turned out to look cartoonish and less serious in the interfaces. They were gradually replaced with more neutral tones and reduced to keep the color scheme simple. The before and after versions of the color scheme are presented in Figure 4.4 to illustrate the iterative evolution. Together with the colors the text styles defined in Figure 4.5 form the base for the visual design language.

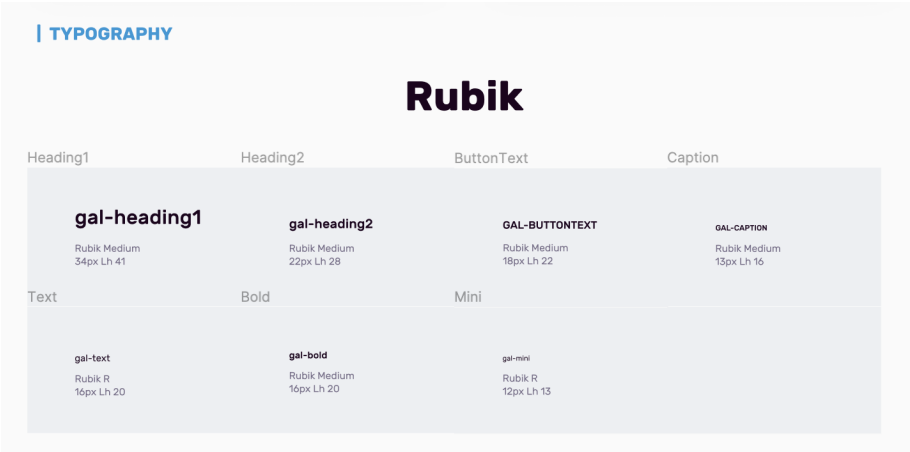

Figure 4.5: Text styles of the visual design language.

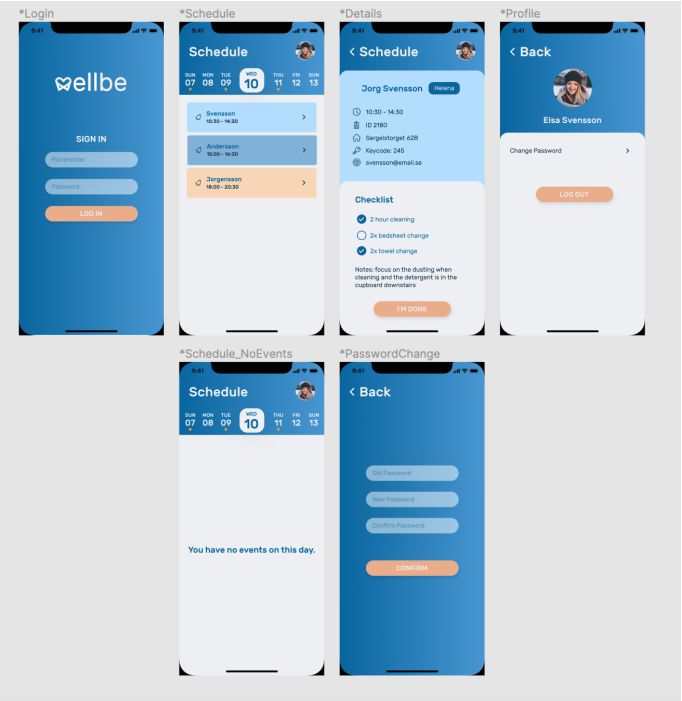

Figure 4.6: First version of the application for cleaners with basic functionality.

Figure 4.7: First version of the application for customers.

The core of the visual design language are the components, which were all taken from the designed application interfaces for two different applications. The first one

was focused on handling the management behind the apartment cleaning service and helping the cleaning staff to stay organised. Therefore, its features focused on managing the different appointments and showing the most important information for each of them such as address, information on entering the building and the cleaning preferences of the customer to help them work efficiently. Figure 4.6 shows a prototype of the most fundamental functionality. Afterwards the work shifted towards a different project because of a change in the priorities of the startup.

This resulted in the second application design, which was a redesign of the existing customer-oriented application for the service orders. The objective of this project wasn't to redefine the user flows, but to quickly and efficiently redevelop it for a web platform and use the new brand identity. Functionally, the application resembles an online shop. The initial, *minimum viable product* version in Figure 4.7 mostly includes common screens, like the browsing section, item details, cart and the checkout process. Some optimisations were also made for larger screens so users could access it from any device, which are visible in Figure 4.8.

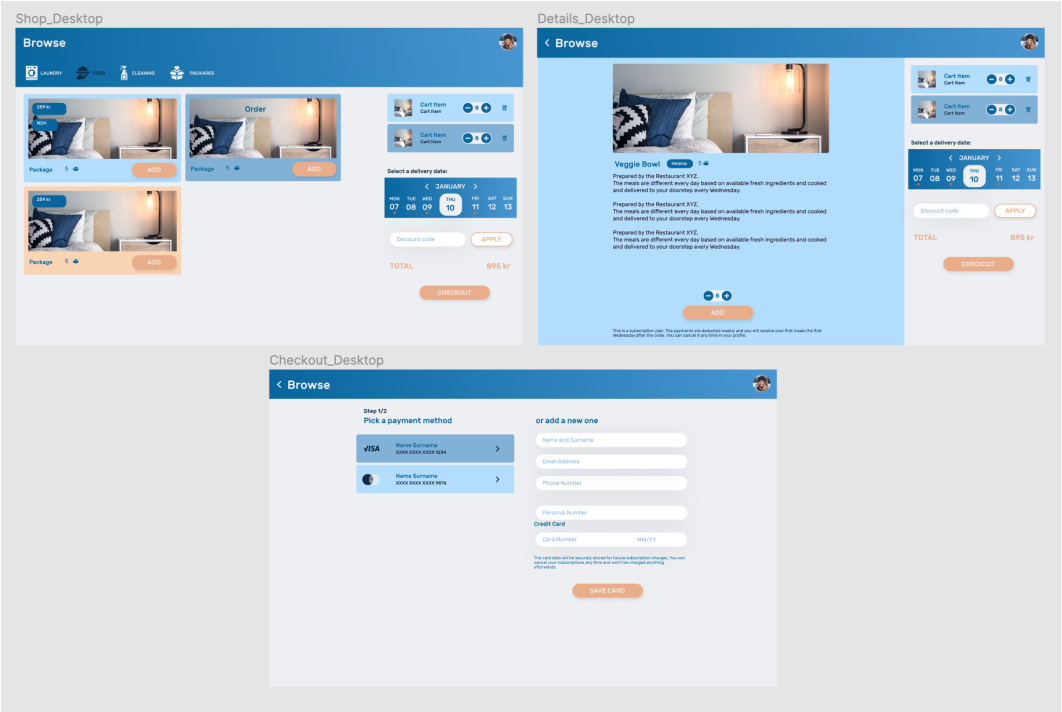

Figure 4.8: Large screen optimisation for some pages of the first version of the application for customers.

It is clearly visible in Figures 4.6 and 4.7 that some components and even entire screens are reused across both applications, which was the purpose of the visual de-

sign language. Even at a very early stage the components were reused instead of continuously creating new elements. The decision of making a design element into a component was based on whether it was a repeating element across the page designs or one of the really basic user interface elements (e.g. buttons). Some inspiration also came from the well known existing Design Systems, like Material Design components and libraries of UI elements like Bootstrap. The assumption of this decision was that if a component recurrently appeared in different component libraries, it was likely a universally useful design element. However, to keep the Design System growth organic I only added elements that appeared at least somewhere in the application pages.

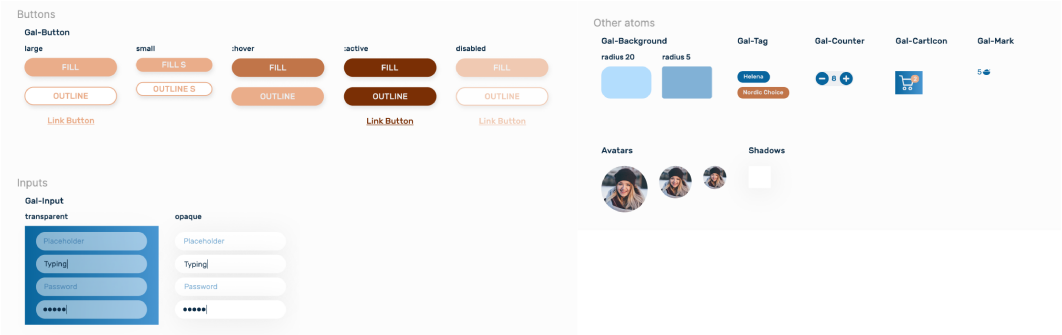

Figure 4.9: Visual design language atoms.

To keep the components structured, they were also categorised and ordered according to the Atomic Design. At this stage, I only defined Atoms and Molecules and left the Organisms, Pages and Templates categories for later when patterns on more complex element combinations emerge. That was done in anticipation of the future design changes after the launch of the application and first round of testing with the actual users. Otherwise, the visual design language would then require more work for restructuring, which would counter the whole purpose of such approach. At the same time such lightweight approach leaves room for creativity and exploration of the designs. However, for better structure the components were also grouped within their categories based on their purpose and use wherever needed. An example is the Navigation group for toolbar and tab components. All of those components (Figures 4.9 and 4.10) formed the initial version of the visual design language.

4.1.4 Component library and application Development

The component library was built to reflect the visual design language with the aim of having the code resemble the designs as close as possible. As established in section

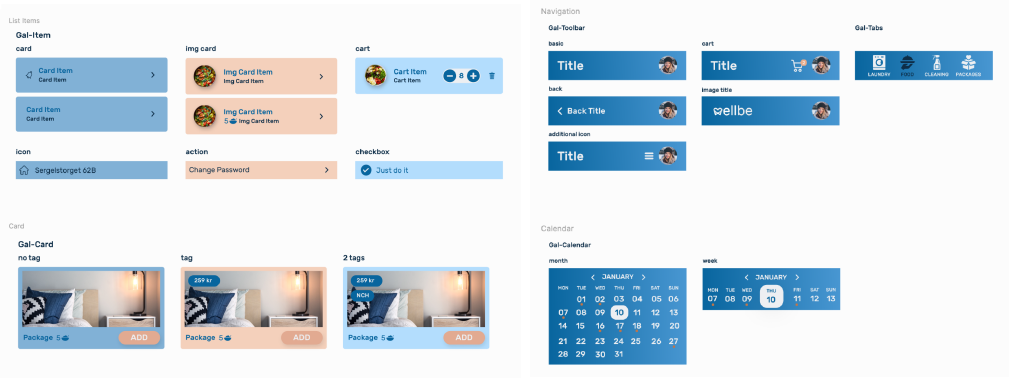

Figure 4.10: Visual design language molecules.

4.1.2, the standard used were web components built in Stencil.js and then exported using a Stencil option for Angular to be used in the actual application. The exporting was done through a published *npm package* that Stencil is also really compatible with as it takes care of the entire setup automatically. Another important concept used was the *Shadow DOM*.

DOM is a Document Object Model used in HTML documents and Shadow DOM is its feature that helps building self-sustainable components (Bidelman 2019). It creates a sub-tree within an element positioned in the main DOM tree, so its children elements' styles cannot be overwritten by the document styling (Bidelman 2019). In practice, this means that the component will always look and behave the same, regardless of where it is being used. Therefore, such practice is very useful to achieve design consistency and forms an important aspect of the Design System.

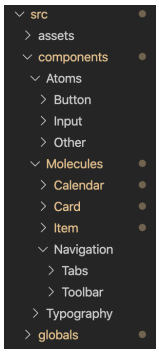

Figure 4.11: The structure of the component library.

The components in the component library followed the same structure as the visual design language. The only difference was that some variations of components that are visible in design all belong to a single component file. Those variations are then

handled in the code. This tree structure is showcased in Figure 4.11. As can be seen there, *Typography* is one of the folders where I decided to test the concept of having the text styles as standalone components as well. This option worked out well as they were really easy to reuse and also made it possible to limit the text colors to those that would suit the designs the most.

```
/** blues **/
--color-blue-dark: #08659E;
--color-blue: #81B1D6;
--color-blue-light: #B3DDFD;

/** grays **/
--color-gray-dark: #0E2E48;
--color-gray: #A8B8C1;
--color-gray-light: #EDEFF2;

/** neutrals **/
--color-neutral-darker: #7A2F05;
--color-neutral-dark: #C16649;
--color-neutral: #E2AA91;
--color-neutral-light: #F4D0BB;

/** gradients **/
--gradient-blue: linear-gradient(85.93deg, #08659E 0%, #4997D2 100%);
--gradient-neutral: linear-gradient(55.23deg, #7A2F05 0%, #C16649 92.47%);
```

Figure 4.12: Color styles in the component library.

On the other hand, the colors were set globally using CSS variables to make it easy to reuse them across all of the components. They were also structured and named the same as in the Figma project with the prefix of *–color*, which can be seen in Figure 4.12. To quickly export the styles, especially for the gradients, I used Figma's Code feature that translates it into CSS styles.

The application development in big part served more as a way to test out the component structure in a real use case scenario and iteratively optimise them for use. Therefore, having the component library development so closely intertwined with the application development was very useful. The main insight resulting from it was knowing what actions and behaviors the components needed to support beyond their visual design. In Stencil the actions are handled through *Event Emitters* that can be listened to and handled in the application. To avoid adding them redundantly, I only added them as the needs arose in practice which kept the process very efficient. Some changes focused on the naming conventions as well, which helped with the general naming of the components and their properties to become clearer over time. Other than that, there were some changes needed in the design as well due to the differences between the real data and the ideal data anticipated during the design process.

In Figure 4.13, there are a few screenshots from the resulting mobile application

Figure 4.13: Application screenshots.

for comparison of the design and implementation. The text, however, was translated to Swedish as the application is launching in Sweden and targeting users of that market for the time being.

4.1.5 Single source of truth and Documentation

To document the Design System and also keep it synchronised I created a Storybook to have as the single source of truth for it. I used the version of **Storybook for html** because at the time of writing, the version for web components did not yet have official support or documentation and was only being tested. This limited the implementation, but there were no major problems. For the structure of the components in the Storybook I used the same structure of Atoms and Molecules as in the visual design language and component library and added the styles section to show the typography and colors. The tree structure of the Storybook is shown in Figure 4.14.

As the Storybook comes with different addons, I used them to document the components and also provide a single source of truth by combining the design and code in the same place. I used a **Knobs addon** that allows you to edit the props in the user

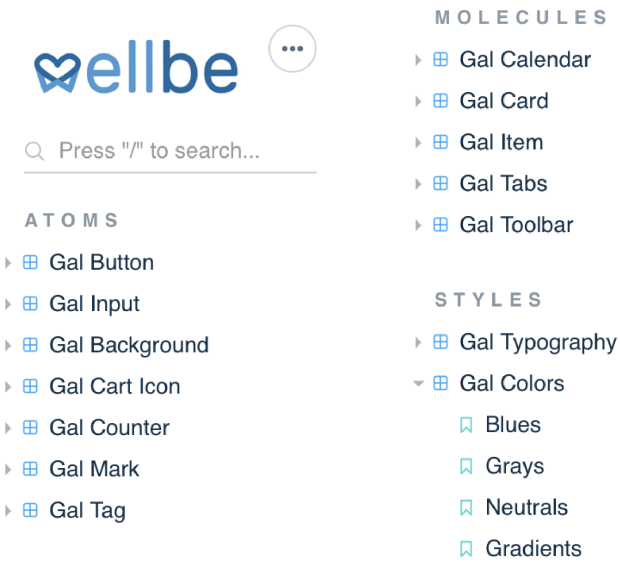

Figure 4.14: Storybook structure.

interface and re-render the component in real time. Combined with the **HTML addon** that generates code snippets based on the value of Knobs it makes it really easy to set the component up in Storybook and then just copy and paste it to the project.

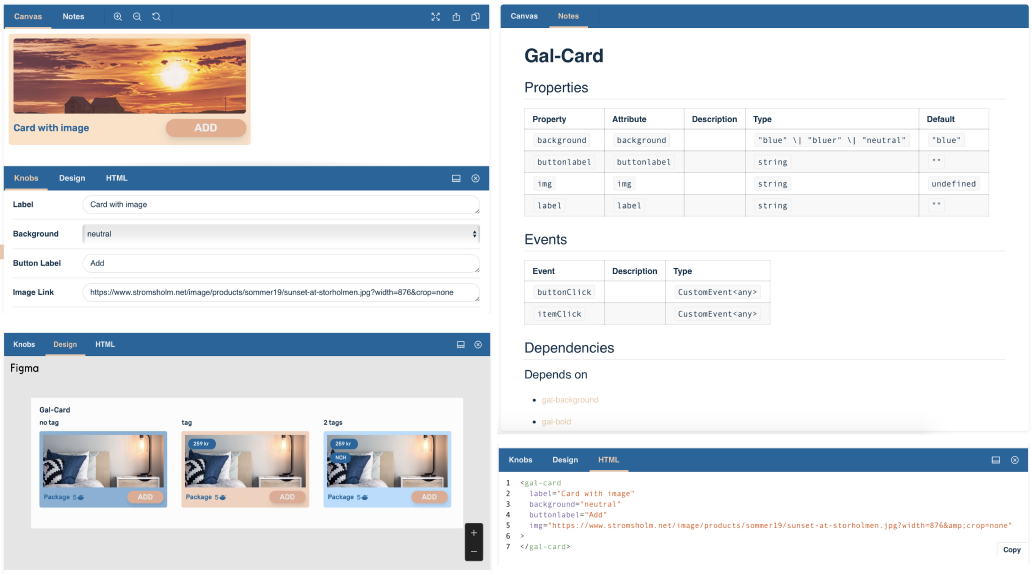

Figure 4.15: An example of a card component in Storybook.

For documentation, I added the **Notes addon** that shows automatically generated Readme files from Stencil with all the Properties and Events of the component. As of now the documentation is kept very basic, but Stencil also allows to add custom descriptions to the file. Finally, the **Design addon** shows how the components look

like in Figma, which makes it easier to stay in sync without having to go back and forth between the tools. Figure 4.15 shows an example of the card component in Storybook, where each of the addons takes up a tab together with the Canvas tab that shows the actual rendered component.

4.1.6 Final Design System

The Design System created through this process is very compact to make it easily maintainable and scalable. The Storybook keeps everything within a single file that can then be accessed by the developer working on a project together with the npm package of components to implement the products. For the designer, everything is kept within the design tool (in this case Figma). The goal was to make it usable for everyone and easy enough to pick up by potential new employees and keep in use for the team.

To determine if the goals were reached it was evaluated through a user test session that took place online, through a video call. The user chosen for participation was a developer with no previous experience with Ionic, Stencil or Angular, who also had never used Figma before, but was familiar with component development in React. The user was given the Figma files with two screens designed using the components (Figure 4.16 left) and access to the single source of truth of the Design System. Furthermore, he was also provided with the setup instructions to start a project from scratch and build the two screens without any navigation or data loading. During the test, he was prompted to think aloud and share the screen to enable a more thorough observation of his work.

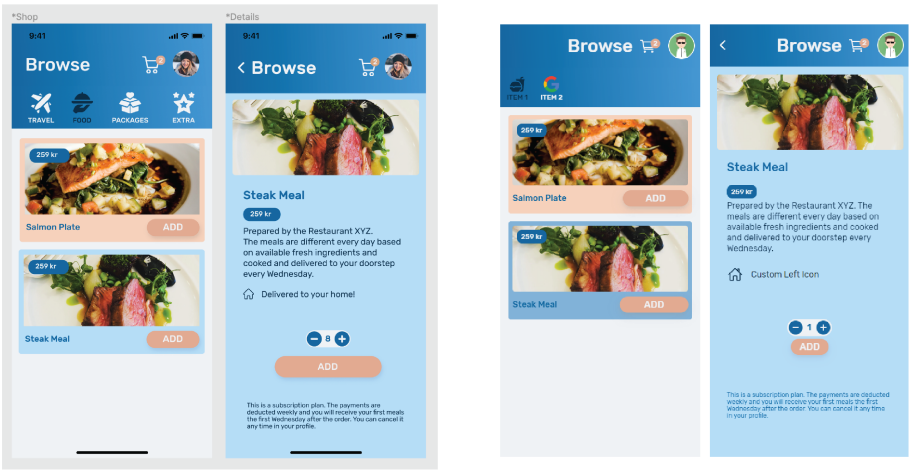

Figure 4.16: User test goal designs (left) and results (right).

The test lasted for an hour and 50 minutes altogether, with the first 15 minutes spent on getting to know the materials and reading the instructions, half an hour for setup and just over an hour for the implementation of the pages. The results are visible in Figure 4.16 on the right. It is clear that not everything is implemented correctly - some assets are missing, some colors and components are wrong and the tabs are using the default data from the example in the Design System.

However, the majority of the design is correctly implemented and it took the user less than two hours to learn how to use the Design System and complete the implementation, which was less time than expected. The major issue turned out to be the setup of the project, partly due to the lack of previous knowledge of Ionic and also because the current setup of custom components requires some specific Angular tweaks that need to be followed precisely. With regard to the user feedback during the implementation, I finalised a set of instructions to be followed by the developers until Ionic provides a permanent fix for this issue. Other than that, another time consuming challenge for the user was finding Figma's export option for the assets, which was likely due to the lack of previous knowledge of the tool.

During the implementation of the components the biggest problem was the lack of documentation on proper data binding to the tabs component and also on the usage of the component events in general. Another major identified problem was no mention of the ability to use Ionic icons in the components that have any, so the user spent a lot of time trying to match the components in the design. All of the above have been added to the Design System before its final submission to the startup.

The user test showed that the Design System is usable for people with the basic development skill set and that it can be used to quickly get started on a project. From the participant's usage of the Storybook it was visible that the different addons served their purpose. Generally, the user used the Knobs and HTML sections most in order to setup and then copy components to the project and didn't use the Design function, which has a synchronisation purpose and was therefore not so useful for this task. The Notes section was mostly used in case of any trouble with the implementation such as in the case of the above mentioned Tabs component. Overall, the results were promising and the performance was good.

Because a long term evaluation of the Design System was out of scope for the thesis, the startup was also asked to provide an impact testimony. The testimony can be seen in the Appendix B and contains the perceived quality of the final artifact. The Design System was very positively accepted by the startup team and will continue to

be used for design and development in the future. According to one of the co-founders, it is fitting for the needs of a startup due to its modularity and will ensure high quality of design going forward. In summary, the startup team aims to use it to increase the brand awareness, iterate faster and improve the user experience.

4.2 Qualitative Analysis of the interviews

In this section I present the outcomes of the qualitative analysis of the performed interviews. First, I introduce some general information about the startups and employees and then present the concepts and themes found in the analysis. In Chapter 5 those findings are tied together with the empirical experience of the Design Science Research described above to answer the research questions.

4.2.1 About software startups

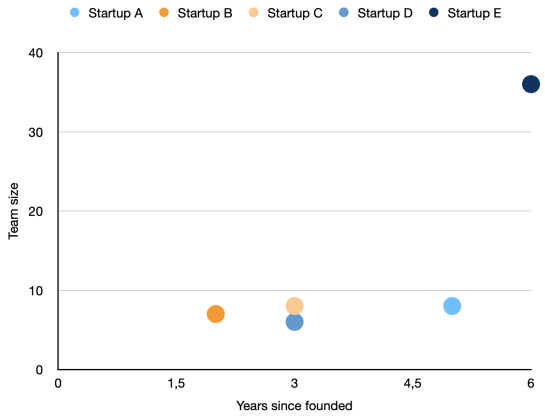

Figure 4.17: Team size and age of the startups.

In the Introduction chapter of this thesis I defined a software startup as up to 8 years old company with no more than 50 employees that has software as part of their product or service. All of the interviewed startups fit this criteria and had either a Web or Mobile based product or service. Figure 4.17 presents the team sizes and ages of the 5 interviewed startups. As is visible, none of the startups are older than 6 years or have more than 40 employees. The youngest startup was founded 2 years ago and the majority of the startups have teams significantly smaller than the limit with less than 10 employees.

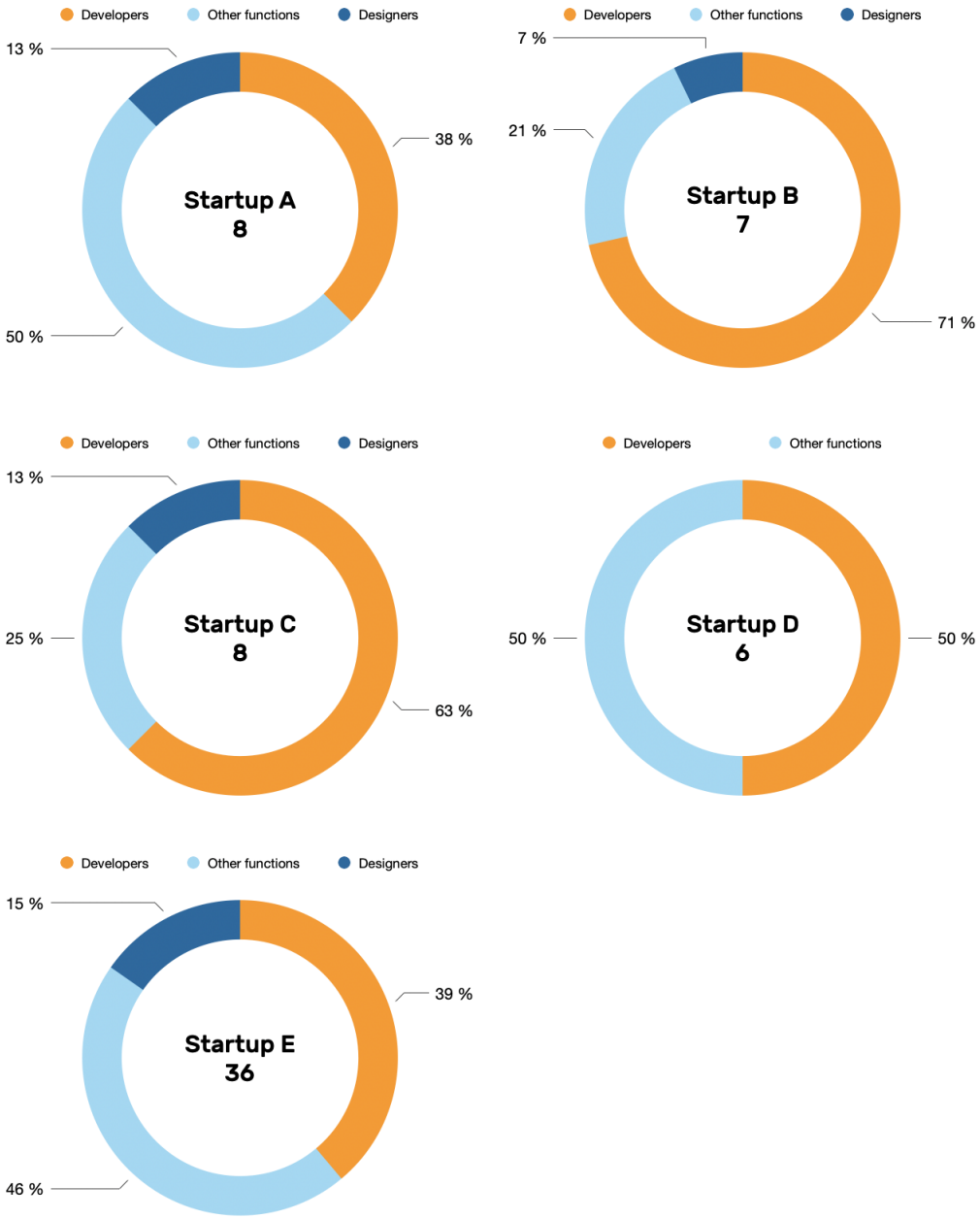

Figure 4.18: Team division into developers, designers and other functions.

To get an overview of the team structure, Figure 4.18 shows how the teams are structured by roles. I differentiate between the people who work in any kind of development, both software and hardware or even data science positions (referred to as *Developers*), *Designers* of the startup and *Other functions* like management, marketing, content development, etc. The prevalence among the teams is that developers take up a much larger portion of the team than designers. It is important to mention that some employees mentioned taking on multiple roles in which cases the employee

would be equally distributed among those roles to give a more precise insight into the team structure.

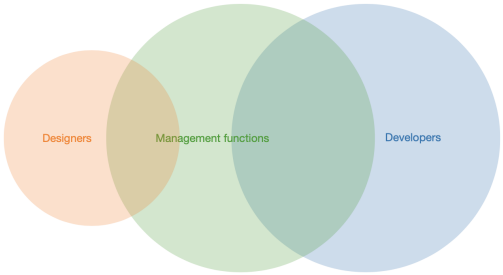

Figure 4.19: Roles overlap of the interviewed employees.

In the interviews, I also focused on three kinds of roles - design related (e.g. a UX Designer), development related (e.g. a Lead Developer) or management related (e.g. founder, CEO or a Product Manager). Since a lot of the interviewed employees had positions that combined multiple of those role types, Figure 4.19 shows how those roles are overlapping for the interviewees. For reference on where the data for the Figure came from, the table 3.2 from section 3.3.1 is repeated in table 4.2. However, the actual positions were replaced with the role types. The interviewees are later on in the analysis referred to with the letter of the startup and number of the employee in the table. For example, the third employee interviewed from startup B would get a label B3.

	Employee 1	**Employee 2**	**Employee 3**
Startup A	Designer	Management	Developer
Startup B	Designer, Management	Developer, Management	Developer
Startup C	Developer, Management	Designer	
Startup D	Developer, Management	Management	
Startup E	Developer	Management	Developer

Table 4.2: Interviewed startup employees roles per startup.

As is visible, there is no overlap between the design and development, meaning that the startups mostly employ people who do one or the other. That could indicate a gap between the two processes with the lack of people involved in both, who could advocate for a better system. At the same time, it shows that in the teams that are already developer prevalent, none of the developers work on design. The assumption forming here is that design is possibly disregarded in the software startups. Nonetheless, this

opens up a possibility for people with design and development skills to potentially be able to solve this issue. However, quite a lot of people who work in either of those also take on a management role or co-founded the startup.

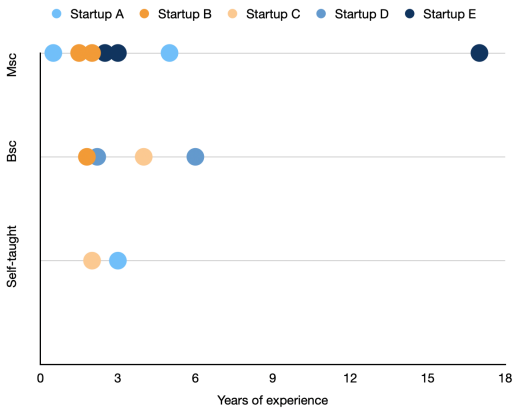

Figure 4.20: Years of experience and education level of the interviewees.

Finally, the interviewees were also asked to disclose their background in years of experience prior to joining the startup and their formal education. Figure 4.20 shows that the employees are either self-taught or have a Bachelor's (Bsc.) or Master's degree (Msc.) level of education and the majority has less than five years of previous experience. However, it is important to note that some of them have stayed with the startup for a while or even since the founding, so the plotted experience does not necessarily match their current level of experience.

4.2.2 Emerging themes of the interviews

In the thematic analysis the topics identified in the conversations were structured and organised based on the themes that were of interest to the research. For better clarity the coded topics were plotted in two networks to showcase their connections. The first one (Figure 4.21) focuses on the different aspects of startups work and challenges. The startup work has been divided into organisational aspects that cover general matters like communication, decision making, budget, etc., that affect how work and changes happen on a daily basis. Other identified aspects are specifically focusing on all design work, the development work and lastly, the hand-off.

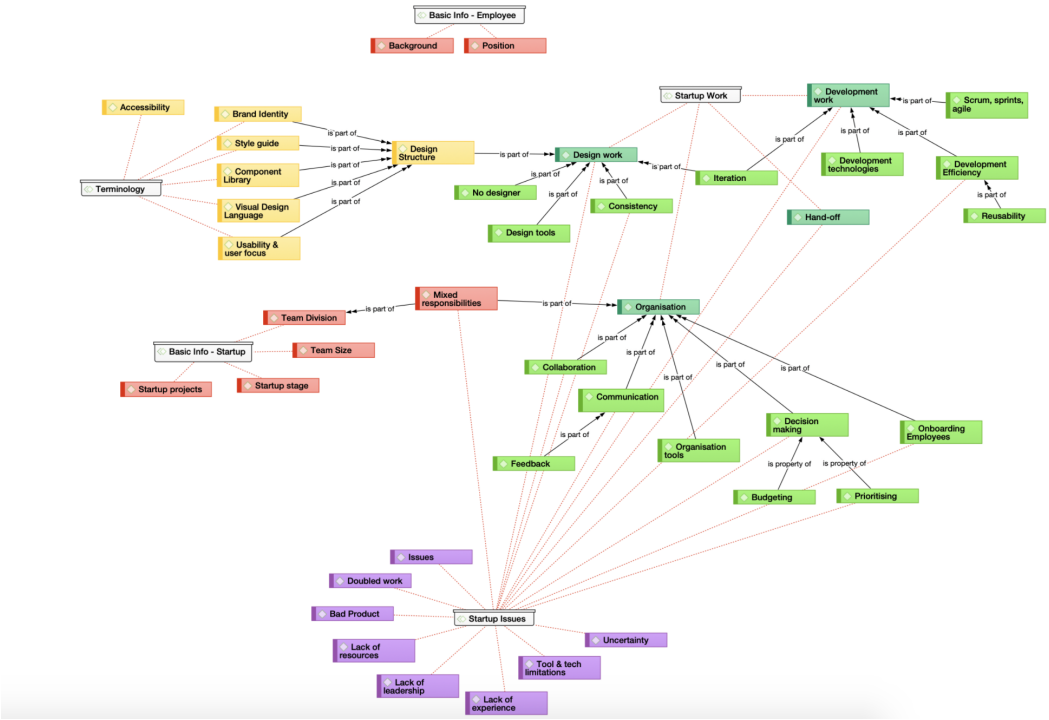

Figure 4.21: The network of identified topics regarding the work processes in startups.

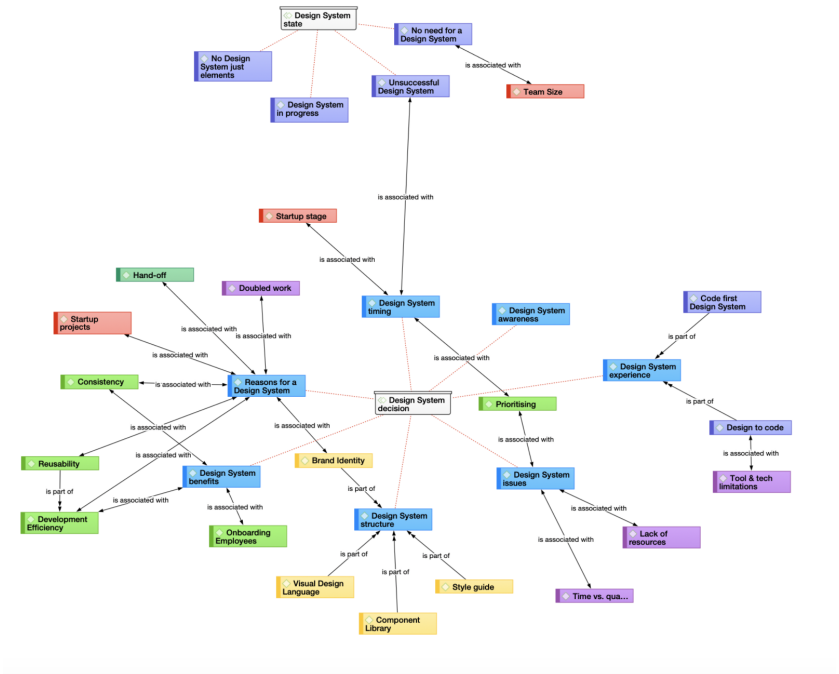

Figure 4.22: The network of identified topics regarding Design Systems.

The second network in Figure 4.22 covers the topics of discussion around the Design Systems concept and the relation between it and the general software startup

work aspects that were established previously. The main identified themes about Design Systems were current state and direction of the software startups and how they perceive and what they base their decisions on in the matter.

4.2.3 Design work at software startups

Especially in the beginning, the design work at a software startup is of lower priority than development, which was already visible in the team structure of the software startups. In this situation it is usually handled by people in other roles. Sometimes it is up to the developers, like in startup C (in the past) and startup D, which still doesn't have a designer in the team. In both cases it led to inadequate design that, especially for startup D, is still a problem today.

> *"The developers who don't have a background in design would be doing design. I feel as though it looked very weird."* (C1)

Another option is to have the design work done by someone with background in design that took on a different role in a startup, usually that of a co-founder. Such was the case in Startups B and E (in the past), where the CEOs took on the design tasks. Compared to having it done by the developers this seems like a better option as neither report any specific issues with the design quality. The lack of designers in the team is often explained as the *lack of need*, which caused both Startups B and D to dismiss part time designer employees.

The tools used for design are Figma (4 out of 5 software startups), that was combined with Sketch in one case, or Adobe XD (1 out of 5 software startups). Some reasons for the tool choice are the features, price and ease of use: *"I use Figma. I used to use Sketch, but I went over to Figma because it's free and it's more collaborative. For prototyping I used to use InVision, but I think it is an overkill at this point because we are a startup."* (B1) Figma is also liked by the developers because it enables good communication with the design team, according to A3. On the other hand, the benefit of Adobe XD is the integration with the other Adobe tools and the rapid rate of updates and improvements.

When reconsidering the tool of choice, the functionality is not the only measure. Time needed to switch between the tools is also considered: *"Everything that has been made so far has been in XD. We would lose a lot of time if we were to change over to Figma, InVision or something else."* (C2) The choice is usually up to the designers and their previous experience and preference, which is reported by Startups A, C and E.

The elements used to structure design across the company are mostly some forms of style guides and or brand identity. However, the formality of the structure differs between the startups. From no structure in Startup D, due to the lack of time and resources according to D2, to having a brand identity for each of the projects along with a general startup brand (Startup E). The rest of the startups are somewhere in between: *"We had our primary colours and our main font and pretty much just like a basic brand identity guide that's not specific to UX or UI design."* (B1) On the other hand, only Startup C mentioned working on a set of ready made components that they named design kit.

The design work is closely tied to the development and usually performed in iteration (mentioned in Startups A, B, C, E). The way it is being described by both designers and developers, it usually intertwines closely with the development work. That is likely as some developers mention that the majority of the testing happens after the implementation: *"After the design process is done, I would go and implement whatever is in that design in Figma. And then we would go and test. And then if we decide something doesn't make sense from a usability perspective, we'd go and re-design and re-implement."* (B3)

Two startups bring up the issues with consistency across the designs, in Startup C the problem was the typography, while Startup A stated it was a more extensive problem they have been dealing with for a while. In general, the startups are beginning to recognise the need for structure. Therefore, some startups are actively looking for solutions, including Design Systems.

> *"That's what I'm working on right now, setting up the Design System because it's also really important to have brand and consistency."* (B1)

Figure 4.23: Key findings about the design work at software startups.

The findings about the design work at software startups are summarised in Figure 4.23.

4.2.4 Development work at software startups

Development technologies differ based on the product. In this case two startups are focusing on mobile applications (B, E) and three are working on a web platform based product (A, C, D). For mobile development both startups decided to develop native iOS and Android applications separately instead of using a framework despite the double work. There are different reasons for that decision, from giving a native feel and using the native components (Startup E) to adapting to the users (Startup B).

> *"We started with iOS partly because our early customers have iPhones, but we are now trying to figure out if React Native is something that we should maybe switch to just because we are still a small team and it's a bit of a waste of our resources to develop two apps at the same time."* (B2)

On the other hand web development is done in frameworks, either with React (Startups A and C) or Vue.js (Startup D). React is the preferred framework, even Startup D chose Vue.js as an alternative due to the issues with terms of use for React not because of the capabilities. None of the startups chose Angular. This is important because frameworks have a strong community support that is very helpful in case of any issues.

In regard to work structure, there is a lot of mentions of agile developments and scrum methodology. According to the interviewees, it helps keep up the progress and accountability of the team. Especially for scrum, they recognise the usefulness of the methodology: *"Scrum is usually very good for things that are new and high uncertainty and that require a lot of trial and error."* (C1) This is likely due to the emphasis on keeping the development efficient. For example, in Startup A they consider it a matter of satisfying the users, while multiple startups value reusability as a way to save resources. Prioritising development speed also goes back towards the design choices, which has the potential to negatively affect the designs.

> *"It also needs to be feasible, sometimes great design would totally do the thing, but you realize we can't build it this way, or, you know, in we could build it this way, but it would take 37 weeks, and if we made these three subtle changes it would only take us two weeks. In this case we're gonna make those changes because it is faster."* (E2)

The key findings about the development work at software startups are showcased in Figure 4.24.

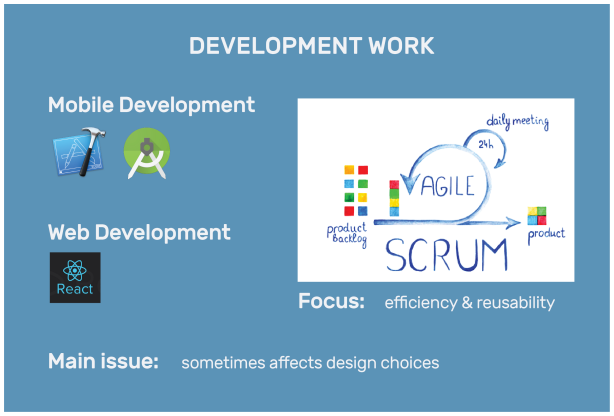

Figure 4.24: Key findings about the development work at software startups.

4.2.5 Hand-off at software startups

Startups handle this step very differently. For startup D, without designers in the team, there is no hand-off as the design is done during development. Other startups have different levels of structure in place from using a separate tool like Zeplin for clarity to basing the hand-off on the communication between the developers and designer(s). The results show that larger teams tend to require more structure.

Startup E is the most structured and also has the largest team. They use Zeplin for the screens and assets delivery in the same place. According to E2, the designs in Zeplin are included within the specification document that provides a source of truth for the developers. The latter like this process, describing that *"it's really useful and Zeplin is quite nice because of the way you as an engineer can interact with the mock up - you can see all the distances and sizes and export assets in the sizes that you need."* (E3) Part of their hand-off structure is also an additional feedback loop after the implementation where designers have to give approval on the appearance. The reason for it is high design standards.

> *"From Figma, this is the part where you can just give it to the developers and they can look at it right here and start developing it."* (B1)

Contrarily, the rest of the startups with smaller teams rely on the tools they use for design. This works because of the sharing ability of those tools, however, the scalability of such approach as the team expands is questionable. It requires direct communication

to resolve issues that these teams are relying on: *"There's sort of like a back and forth between me and our CEO, who's also our designer."* (B3) There were no conclusive insights into how time consuming is such practice, but it could become chaotic as the team grows.

Some issues that happen with hand-off are the lack of clarity on when something is finished and ready to be implemented (Startup B) or misunderstandings between the designers and developers (Startup C), both of which could indicate the lack of structure. There are also problems when trying to push new features out too quickly without regard for how the real data affects the interface.

> *"The main problem that we encountered was that we were developing the features so quickly that he [the designer] was providing wireframes, but without always thinking about all the technical issues that we can have for implementing. For example, if you create a button and then there is a bigger text inside than you haven't predicted."* (A3)

Another issue is the collaboration between the developers and designers, where they sometimes do not take each other into account, which slows down the progress. As E1 reports, it helps to involve both designers and developers throughout the process.

Examining the overall satisfaction with the hand-off process, some people are happy with it like Startups B and E, while it is a pain point for Startup A. *"We were always kind of frustrated in this interaction and still haven't found the way to really merge the two. There's always a process of weekly hand-off and then we need to discuss things. And then it's always looking different on the website than on the designs because we have to make some compromises."* (A2)

The findings about the hand-off at software startups are summarised in Figure 4.25.

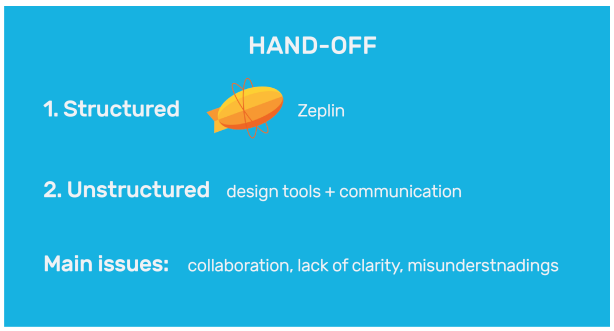

Figure 4.25: Key findings about the hand-off at software startups.

4.2.6 Organisational aspects at software startups

The organisation is important for the startups work and the startups are using similar tools to organise work. Slack is used for communication by all five startups, Trello (1 out of 5) or Atlassian JIRA (4 out of 5) for the task management and Google Suite for other needs. There are cases where the design and development teams do not use the same tools to plan their work: *"We started using JIRA to divide tasks but never for the design team. For them I wrote plans in Google Docs to be honest.* (A1) A lot of communication and collaboration also happens in person in the office or through video calls.

Part of the structure are also the responsibilities of the team members. As was mentioned before, one recurring theme was that the roles of the team members are not so clearly defined in the startups and each member usually takes on whatever is needed in the moment. This was an emerging pattern, especially in smaller startups and was not mentioned in Startup E with a much larger team. However, it was mentioned more often by the people that co-founded the startup than people that joined later on as employees.

> *"As in many startups, this [the role] is just a title that doesn't really mean anything because what I do is pretty much everything"*. (B2)

In terms of making the decisions, the startups mainly look at long term goals. Besides fitting the purpose, the technology and tools are supposed to be reliable, scalable and long lasting. Startup E specifically emphasised that reliability is preferred to trends, while descriptions of other startups indicate that they also do not easily switch if there is no good reason. The decisions are usually also made as a team not always just on a managerial level. *"It's mainly based on who will use it most and who wants us to get it in the first place. That person is responsible for it so the decision-making is not centralised where I would go and decide on everything."* (D2) However, there are things to consider, like the return on investment or if the difference is big enough to justify the change. All five startups also emphasised the need to take time to make good decisions upfront as it pays off later.

A lot of interviewees talked about the difficulty of prioritising. *"For a while we had a problem setting priorities, especially as the CEO didn't know what was the highest priority so everything seemed urgent to him."* (D1) Set priorities dictate team expansions, workload organisation, whether to focus first on design or development, etc. In the last case, development is usually first to ensure the product is built. Startups A and B report using a 80/20 rule to deal with it and disregard perfection in order to keep things fast.

Another aspect of decision making is also having a budget. Here, the startups vary on a range from never paying for anything (Startup A) to using a free version and seeing if the expense is necessary (Startup B). In majority (3 out of 5 startups), they specifically mention having a budget and being willing to pay for something of value. *"We use whatever is going to work best for the team. So budget is not really a concern often. Most of the tools that we use are either very cheap or pretty reasonable."* (E1)

Figure 4.26 represents a summary of the organisational aspects at software startups.

Figure 4.26: Key findings about the organisational aspects at software startups.

4.2.7 Challenges for software startups

Some challenges were already mentioned as part of the other aspects of startup work so this section focuses on the specific issues that were brought up and haven't been mentioned yet. For better clarity the identified themes are shown in Figure 4.27 together with their groundedness scores (amount of times they occurred in the interviews).

Figure 4.27: Identified challenges in the startup interviews.

As seen in Figure 4.27 above, the most mentioned is the lack of resources, which was reported by all startups but one that also has a much larger team. It is an obstacle

for progress and contributes to the pressure for the team members: *"Just feeling like you don't have enough resources to do things and having to take shortcuts and then not know if it is the right thing to do."* (B3) This is a problem from the start and the best ways software startups found to deal with it is by learning to prioritise and reusing as much as possible.

The next most common issue is having to choose between saving time and quality. Within all of the interviewed software startups there is reports of feeling the pressure to deliver quickly at some point and having to take shortcuts to manage the workload. The perception among some interviewees is that if they aren't fast enough, they won't make it as a startup. However, over time the majority already came across the realisation that sacrificing quality for speed does not work and usually ends up wasting more time in the end.

> *"We were building like 15 views and so many different components and about this rushing through things I realized kind of afterwards that it's a trade off. ... And so afterwards we had to refactor everything and it has taken a lot of time to do it."* (A2)

Reflecting back to the Figure 4.20, the majority of the interviewees had less than five years of experience. Lack of experience is also reportedly a problem in 4 out of 5 startups, especially since some of the interviewees have not worked at a company full time before. *"It can be because we were all very junior in what we did and the product was not well-defined. ... We were all almost discovering the wonderful world of building an application."* (A3) Nonetheless, due to the smaller team sizes they still have to take on responsibilities that would likely be filled by more experienced people in larger companies. This might work in some positions, but multiple statements show that it presents a problem, especially in the management area and decision making.

> *"The biggest thing is that we have a lot of people who are either really new at the their career or have never really been anywhere else so they don't have that sense of comparison you get over time. They found comfort in one way that works and big changes are very frightening to them. Now they [the startup] are correcting this by hiring a lot of people to manage the different teams with more experience so they recognise they need for that depth in the management bench."* (E2)

There is also an issue of doubled work. Sometimes it happens due to the organisation issues or poor work structure (Startup B), other times it comes down to the design problems like infinite redesigns without a satisfactory outcome (Startup A). It can also

happen because of the coding mistakes of the system architecture setup (Startup A). *"You start building stuff on top and you end up with the tower-like thing and then you just figure out that if you want to remove the brick at the bottom, you have to completely reconstruct the tower."* (A3)

Sometimes the challenge is dealing with the limitations of the tools and technologies. Those happen both within design (e.g. Startup C struggled with complex designs in Adobe XD) and technology (e.g. Startup B faced issues from the lack of consistent typography options in native iOS development) and have been brought up by team members in every startup. It is indicative from most answers that the majority of the limitations are not extensive enough to justify the time consuming switch to an alternative. Such changes would usually happen if multiple team members are experiencing the issues and only after a thorough examination of the alternatives.

Two of the startups mention challenges with having a bad product as well, mostly due to not understanding the users or meeting their needs. *"We still have some counter-intuitive design where we have very weird concepts so the users still struggle with it and that is just one of such choices we made along the way."* (D1)

The last two issues are not so common, the first one is the lack of leadership. For Startup A it was a problem with the general startup decisions, while it emerged due to the lack of a product manager in Startup E. *"We've suffered, I think, from the lack of a product management team. When it comes to defining specifications and making sure that a feature is completely thought through."* (E1)

Finally, uncertainty is a problem for the team members. *"It's freaking hard to start a company and it's very hard to also be part of the early founding team. Most people that we worked with, they haven't been able to keep it up for a long time. I think that's probably one of the biggest problems."* (C1) Some interviewees also comment that they feel the need for stability, which is why it can be hard for them to persevere in a situation such as a startup journey.

4.2.8 Design Systems awareness and experience

I asked the participants to explain terms related to design and Design Systems to see how consistent is the terminology understanding, especially among the people with different backgrounds. The terms included were style guide, brand identity, component library, visual design language, usability and accessibility.

Generally, the understanding was good and most of the terms were explained cor-

rectly. Brand identity is associated with values, logos, messaging and understanding of the company. For a style guide, every participant asked talked about colors and typography. An interesting case was one of the developers who did not know the term, but explained it through reasoning. This indicates that style guide as a term is intuitive enough to be recognized.

> *"I think from the coding perspective, like the style in which you write code. I don't know how this would affect design. Maybe in terms of staying consistent across different types of colors or fonts that you use across different components of a product. So style guides would just be a set of rules."* (B3)

In terms of the component library, the understanding differed a little between designers and developers as each explained it from their perspective, however, three people also acknowledged both options in their definitions. The design perspective defines it as a design document with all the component variations and states that you use when creating user interfaces, which is closer to the visual design language or a design kit. In contrast, the developer definition is more like the following: *"A button is something that we use a lot of places, headings we use a lot of places, paragraphs we use a lot of places. And so a component library is simply a collection of these elements within a system that allows you to re-use them."* (C1)

Visual design language was the only term where some people did not know how to explain it and that no one explained according to the expectation. People mostly related it to the style guide and or brand identity, but didn't know how to explain the difference. *"To me that's very related to the style guide. The style guide is the document that formalizes beautiful design languages. But I think there's a lot of things in visual design language that don't necessarily make it into a document. I tend to use that term often when if we are creating a new feature or a new game when there might be some things that you want to keep consistent."* (E1) Another repeating expression was "design decisions". There was also confusion on the difference between the Design System and the Design Language terms, where Material Design (which is a Design System) was mentioned in both definitions.

On the contrary, both usability and accessibility were also mainly correctly explained, usually by mentions of *"good user experience"* for usability and *"impairments"* for accessibility.

Awareness about the term Design Systems is significantly lower than the rest of the terminology, yet still higher than expected from the background research. In Figure 4.28, the level of awareness is presented based on the role of the person withing the

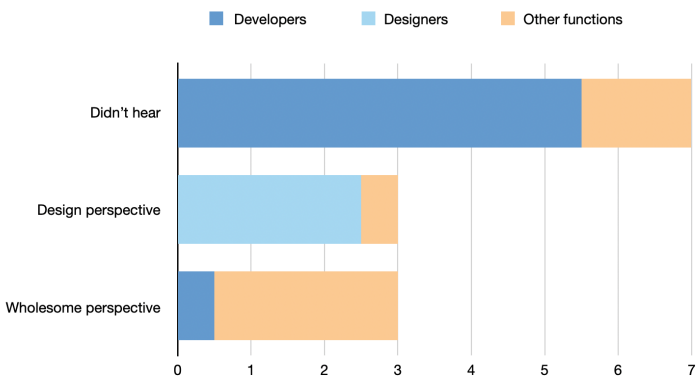

Figure 4.28: Awareness of the Design Systems term based on the role.

company. Since some people take on a mixed role they were counted as representatives of both equally (e.g. 0.5 to each of the categories). It is clear that more than a half people have not heard of Design Systems and didn't know how to explain the term. The majority of that share are the developers. All of the interviewed designers knew about the term but defined it only from a design perspective.

> *"It's something that's similar to a style guide and branding. The Design System is more complex, it's a very specific guide of how everything is made. Style guide would be implemented into a Design System. A good example is Google, with their Material Design, which contains not only colors and cards and visual components, but also shows how they are to be used."* (C2)

An interesting finding was that people who actually understood its full potential were mostly in the management roles. Their definitions were focused on the potential of this approach for the startup: *"In essence, Design System is the next step in our quest to develop user interfaces faster and more reliably. You can work with so many designers and developers, but at some point they become out-of-sync. So Design System is a structured approach to creating user interfaces that synchronizes design efforts with developer efforts."* (C1)

Furthermore, only one of the interviewees (E2) has had previous experience working with a Design System. Others sometimes used similar approaches for personal projects (A1) or specific design to code tools for faster development (E3), but not the entire system. This is likely due to the limited amount of experience noticed among the participants. E2 reports working for a large company that had a designated Design System team setting the design direction and another company where designers took

lead on the transformation to a Design System due to the issues with consistency of the products. *"The Design System team was just getting started and it was sort of bootstrapping itself, it was just some designers who noticed that the force of entropy was driving the experience in very different directions. That's a great example of a company that was around 250 employees at the time but badly needed a Design System because of the structure of the company and the nature of the product."* (E2)

The key findings about the Design Systems awareness and experience are available in Figure 4.29.

Figure 4.29: Key findings about the Design Systems awareness and experience.

4.2.9 State of Design Systems in software startups

Out of the five software startups, two are in the process of having a structured Design System and three are either not aware of the approach or came to a conclusion that it is not necessary and useful for them. The latter was emphasized by Startup E, the oldest and largest startup of the five, where they believe that using separate elements (e.g. a style guide) works well. Further reasoning for the decision is the way they build their products with a lot of design and brand autonomy. *"I think it's about two things - there's a product complexity angle and there's a team complexity angle. On our product, it just doesn't have enough surface area and probably never will because we can't copy patterns across products or across platforms. ... I just don't think it would accelerate anything for us."* (E2)

Startup A follows the same approach because they attempted to have a more structured process in the direction of a Design System in the past, which did not work. Similarly to the Design System built within this thesis they implemented a Storybook, but everyone kept creating the components directly in the code to make the process faster.

The lack of time and consequently adoption was supposedly a source of frustration for a while before disregarding it. Startup D also did not consider Design System approach due to the lack of awareness and they base their design mostly on the existing Material Design components and recommendations.

Contrarily, startups B and C are working on having a Design System at the time of writing, mentioning that it is a big task and takes a while. Their approaches differ between design first and development first implementation, which is based on their needs and the biggest issues that led them to pursue this technique in the first place. Startup B is focusing on the design to improve the brand recognition and consistency, while Startup C has goals to improve the reusability and efficiency of the development process.

> *"The answer to how we can make something reusable and useful for our main projects, even when we're working on contract work became Design Systems, a more structured approach where we have basic components already up and running. That allow us to develop faster for the contracts while also making sure that the components could be reused for our own user interfaces."* (C1)

The differences here solidify the assumption that a Design System is a need-driven approach and is highly individual based on the characteristics of the company. Various experiences indicate that it is not generally applicable to any software startup, but also give insights into what should be taken into a consideration when deciding. In conclusion, all of the key insights about the state of Design Systems in the interviewed startups are shown in Figure 4.30.

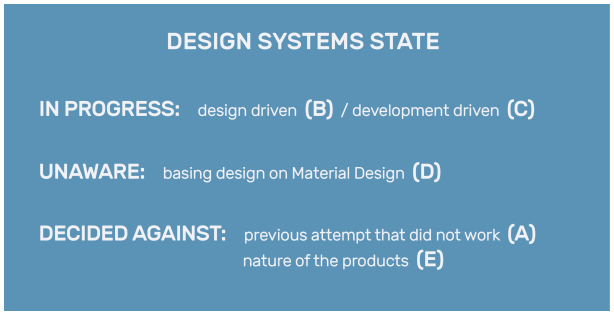

Figure 4.30: Key findings about the Design Systems state.

4.2.10 Design Systems structure and timing

The following section is focused on comments of the interviewees who either had personal or professional experience with the Design Systems or are currently undergoing

the development process of one themselves. Regarding the structure, the opinions are divided. A1 considers it best to keep it light and hybrid and continuously add to it, while C1 sees a combination of the brand identity, visual design language and a component library that Startup C is developing as the best approach. According to B1 and E2 it is more important to keep improving it as you go and add things based on the need.

> *"I think it's too slow to do everything. ... If you have a small team I think you should have more of a hybrid thing, like a small version of a Design System. I think it can be typography, colors, so covering the basic concepts and for example icon set, buttons and forms."* (A1)

Three people spoke on the appropriate timing for a Design System and all three agreed that it does not have to be right away in the beginning but can also come later, after shipping the product. *"I feel like somehow you first have to create a bunch of screens and then you will start seeing the patterns."* (B1) However, C1 adds that it can be useful to keep it in mind from the start, which can help you be more efficient. The most important outtakes are visible in Figure 4.31.

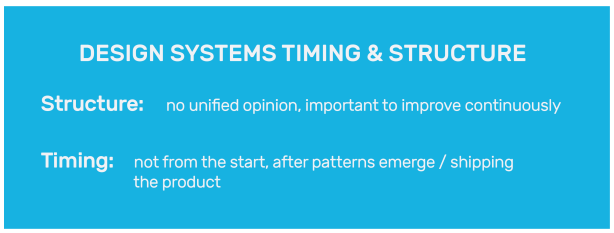

Figure 4.31: Key findings about the Design Systems structure and timing.

4.2.11 Design Systems benefits and drawbacks

> *"We need a more structured way of communicating between the design and the development."* (B3)

Structure is one of the main goals driving the decision of startups to build a Design System. They perceive its benefits in having a system in place that will make other processes more efficient. Another reason for it is providing consistent user experience across products, especially as the team expands. For development, the reasons are reusability of code in a component based approach and engineering efficiency. There is also an unexpected experienced side effect of having a Design System for the on-boarding process: *"A great side effect was that we could use it to give inexperienced*

developers meaningful work that was going to be implemented in the real products which gave them the satisfaction of contributing." (C1)

Due to the lack of experience with the Design Systems overall, there is not a lot of mentions of issues arising from their use. Employees of Startup A that attempted to build it before along with E2 agree on the time and work required to make it work and then maintain it. *"I think it [Design System] is a lie for startups in the very early stages because it's so much overhead in the beginning that you lose focus and it's just gonna be a weight that you carry with you."* (A2) A3 also mentions that despite having potential it still requires both designers and developers in the team to upkeep it, which is a disadvantage for the smaller teams. To summarise, main insights about the experienced benefits and drawbacks have been presented in Figure 4.32.

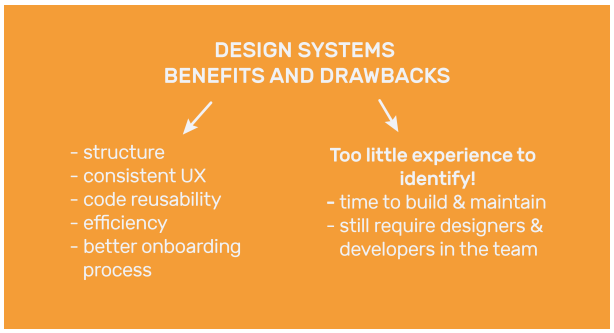

Figure 4.32: Key findings about the Design Systems benefits and drawbacks.

Chapter 5

Analysis

In this chapter I apply the results to answer and explain the findings in regard to each of the research questions.

5.1 What is the level of awareness about Design Systems among software startups?

What are the current practices for design and development among the software startups?

According to the structure of the interviewed software startups and their responses, currently higher priority is given to the development process. Startups choose to hire developers before designers or even choose to not have any designers in the team at all. In such cases, developers take care of the design during their usual development work. It is also commonly more important to be fast than adhering to high design standards. For example, if given a choice between having the perfect design that proves to be complex to execute or a more standard design that is faster, the latter is chosen. However, that does not necessarily mean that the software startups compromise on having good design, as they show awareness on importance of the user experience.

It is also common that the roles are not as clearly divided as in the large companies, meaning that a person in a CEO position might be the team designer as well (that was the case in two of the interviewed startups). However, this kind of intertwined functions usually combine either design or development with different management functions and not the two together. In majority of such cases the reason is also that the person is a

startup co-founder and is at the startup from the beginning so it is natural that they would take on other responsibilities based on their background. Such an occurrence can have both positive or negative outcomes.

For the design work, most startups use Figma as their main tool and usually structure the basics with elements like style guides and or brand identity documents. Sometimes they also include component design, but they feel the need for better structure. The work usually proceeds in iteration and is closely tied to the development. The developers use frameworks, like React or Vue for web development, while most still use native development practices for mobile development. However, they are considering frameworks to make the process more efficient and develop both for iOS and Android at the same time. Agile methodologies are used for the development due to the focus on the efficiency and speed. Reusability is also becoming more important.

Hand-off depends on the team structure, where larger teams take advantage of the hand-off tools like Zeplin, while small teams rely solely on the design tool like Figma and communication between the designers and developers to resolve any issues. Besides, the software startups also use a lot of tools for the communication and task management to structure the work. The latter is planned according to the long term goals for the startup, which affect the decision making as well. Decisions, especially on the tools and structure are also strongly based on the team views and not always up to one person, though that depends on the software startup.

Which challenges are the software startups facing?

The biggest challenge seems to be the lack of resources. Teams are small and they struggle with the slow progress and feel like they have to use shortcuts and compromise the results to manage the workload. This issue usually leads to a challenge of having to decide between the speed and quality, where prioritising speed often leads to issues later on, commonly it results in doubled work. Doubled work also appears on other ends such as organisation issues or not taking the user experience into account when building the product. Some doubling of the work also happens due to the lack of code reusability.

Another problem is the general lack of experience in the team members that are still taking on responsibilities more suitable for an experienced developer or designer. With shortage of designers across teams, they also struggle with the design consistency and prioritising between the design and development. Sometimes the challenges are

the technology and tool limitations, which are hard to resolve, as switching can be time consuming. The process of hand-off and general collaboration between designers and developers is one more challenge for the startups. There is a lack of structure and clarity which leads to misunderstandings, for example when a design is ready to be implemented. It does not help that sometimes startups are not using the same tools to manage those two teams as there has been an example of designers using Google Docs and developers JIRA for the task management. In general, this leads to a lack of synchronisation between them and burdens the communication.

Finally, there are some basic issues that were not the focus of this research but should still be mentioned. One is lack of proper leadership that damages the startup journey and can put if off track and another one is uncertainty, which is stressful for the employees and puts additional pressure on the team to perform well.

How familiar are software startup employees with the Design Systems concept?

In general, the interviews show that general awareness about the terms related to the Design Systems, such as style guide and component library is good overall. However, the awareness on the term Design Systems itself is significantly lower (less than a half of the participants) and the understanding of the term differs a lot. This is likely due to the lack of one definition and various interpretations that can be found in literature. Such insights show the need for better standardisation and resources on the topic. However, it is important to point out that most participants that didn't know the term showed interest in learning more once it was explained.

Consequently to the low awareness, experience rate of working with a Design System in any kind of a company is very low. Although a few interviewees mentioned using similar practices in their personal projects, only one out of 13 worked in a company with a Design System and even that was a large company not a software startup. This individual was also much more experienced than everyone else so this might be due to the lack of experience overall.

Despite the lack of awareness, software startups are deciding to build a Design System. Out of the 5, two are currently in the process and one already attempted to do it before, while only one of the startups didn't come across Design Systems at all and one made a decision to not pursue this approach. This shows that software startups are looking for a better way of structuring their work long term and are willing to put in the work needed in order to get a chance of the benefits of a Design System.

5.2 What are the benefits of a Design System for software startups?

Based on the current practices and challenges that the startups are facing there are some specific benefits of having a Design System. First, a Design System could contribute to the much needed structure for the startups, especially in hand-off process and collaboration. As it involves both designers and developers, it could be used to synchronise their efforts, improve the communication and help the startup achieve the goals. Although some of the benefits of a Design System take some time to become apparent, according to the interviews the software startups base a lot of their planning on the long term outcomes. Therefore, it would be feasible to put some effort into resolving current issues and also ensuring that any team expansions do not disrupt the processes.

From the reports of the large companies the common outcomes of a Design System are more reusable code, increased development efficiency and consistent designs across products and all of these are both valued by the startups employees and common issues for them. More efficient development could help them compensate for the small team without having to compromise quality for speed and reusability could lessen the repetition and doubled work. With more consistent designs the user experience could be better and could consequently lead to a higher success rate.

Having a good system in place would also make it easier for new or less experienced employees to perform well and get acquainted with the company practices. As experienced by one of the startups, even having a very early version of the Design System with a component library made their on-boarding process for the new developers much better, as they could start by developing components instead of working directly on the product from the start.

Since the startups might have to quickly change a lot of things in case of a pivot or a new product direction (as happened during the internship that was part of this thesis), it is also important that the structure supports such rapid changes and twists. Design Systems proved very useful in this case, because the same components were reusable in the new project so the entire process was very smooth and time efficient.

To sum up, there is a lot of potential in this approach for the startups, which does not necessarily mean that every startup should have a Design System, but that it should be considered as an option. Another important fact is that the execution of the Design System is the determining factor in its success and usefulness, so if used it should be taken very seriously and backed by the entire team.

5.3 How should a software startup build a Design System to maximise its potential?

How should the Design System be structured and what elements should it include?

Due to the lack of resources, the Design System should be kept simple, lightweight and only include the necessary elements. That will make it both easier and faster to develop and help it work better as it will be relevant for the work instead of being cluttered and complicated. It might be possible to focus more on either design or development in the beginning based on the company needs to ensure that it serves its main purpose.

Since most of the startups already use elements like style guides and or brand identities, it would be the most useful to use them as a base of the Design Systems along with any other elements that might have been used previously. Otherwise, if there was no structure before, a good guide is to include some basis for the design in accordance with the brand, usually that would be the styles (e.g. colors and typography). Besides that, also needed is an element for code components like a component library or a pattern library that should be somehow in sync with the designed components (visual design language, design kit, etc.). The above approach was used in the empirical part of this research by combining a brand identity, visual design language and a component library and it worked well. It was efficient and not too time consuming to develop, as it took less than 6 months of one person work together with the development of a functional mobile application.

To structure the components I used the principles from the Atomic Design and it resulted in a very orderly system, which was synchronised within a source of truth document that included both designed and coded components. Therefore, Atomic Design is recommended at least as an inspiration, because it can give really useful insights on how to compose the components to maximise their potential. Another aspect that really positively affected the process was using the Shadow DOM feature in the component development. That way, the components had separate structure and styles making it really easy to combine and use them alongside.

What would be the best process to construct the Design System?

The first part of the process is making the decision about implementing the Design System. According to the insights from the interviews about taking the team views into

account in making such decisions, it is possible for anyone within the team to take the incentive and advocate for a Design System. The interviewee with experience of Design Systems even mentioned that in one of the previous companies, the employees took the initiative with it. Figure 5.1 lists some important decision points before implementation. The first one is to decide on the focus of the Design System, whether it should prioritise the design or development perspective or none, and determine the structure (which elements to include).

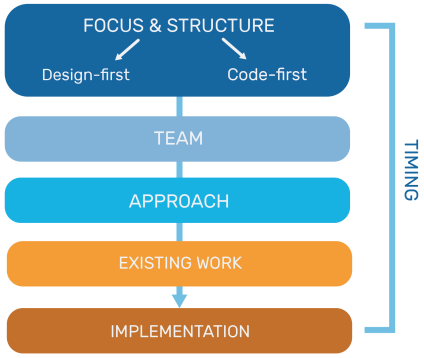

Figure 5.1: Key decision points of the Design System implementation process.

Regarding the team to work on it, it makes sense to keep it small. That way most of the resources are still driving progress of the software startup and it also gives more flexibility for the back and forth work. However, especially if the person or the team in charge is not very experienced, supervision and review of the work is necessary to ensure quality of the end outcome. In the case of this internship, even having only a single person with both programming and design skills working on it was productive and efficient. This is further supported by some interviewees mentioning the work of a designer with some technological background was much easier to work with as the designer already thought of some technical considerations in advance.

The approach to take is based on the focus of the Design system, whether it is code or design focused. In this case the reverse approach worked really well as the focus was mostly on using the Design System to improve the development efficiency and reusability. The component library was developed along an actual product development and it kept the process fast and useful to the overall startup work. The final components were concise and relevant without any unnecessary elements. For a more design focused Design System it might be better to keep it as a side project.

Finally, to save time it can be useful to look at existing Design Systems before implementation. There are a lot of examples of successful Design Systems available online

that can save time and prevent mistakes due to inexperience. The implementation should also take place in iteration to allow the Design System to continuously evolve.

It is also important to take into account the timing. The Design System might be too heavy if the startup is at a very early stage. What was very useful in this empirical case was having a clear brand in place that was partially determined taking into account the feedback from the test users of the minimum viable product. Insights of the interview participants also suggest that there should be a basic idea of how things are supposed to look like and what is the company message. Depending on the software startup, this stage might happen during or after developing the first product.

Which tools and technologies seem to be most useful for a software startup Design System as of now?

For the design portion of the Design System, Figma seems to be the best choice right now. This is based both on the feature comparison and actual experience along with the insight that most of the startups use it already. Figma has a really good system for building and structuring components that can be used in other projects afterwards. However at the time of writing, other tools focused on the translation of design to code, such as Framer, are emerging. Such tools could replace the design tools by automatising the hand-off process completely.

From the development perspective, React is the most commonly used. This is especially obvious with the prevalence of React components in the available Design System examples. There is also a lot of resources available for React and even startups currently using native development technologies for mobile development are considering it. In this use case where Stencil components were built for an Ionic Angular project setup, the biggest issue was both the lack of compatibility with some other tools and lack of community support and resources online that made the development tougher.

The single source of truth and the core of our Design System was built in Storybook, which provided a really good solution and integrated both with the visual design language and component library with little effort. Storybook seems to be the optional tool for code based Design Systems, while Zeplin might be a better choice for the design focused ones. Zeplin is also generally used by larger teams as it requires a bit more effort. However, for very detailed Design Systems, both Storybook and Zeplin can be used together as well.

As most of these tools have an available free alternative and are also not too ex-

pensive otherwise, the budget should not be an issue and the implementation should not exceed a reasonable price. The interviews also showed that the startups, though mindful about spending, usually have budget in place for such expenses anyway.

Chapter 6

Discussion

This is the point of reflection on the research of this thesis. I take a look at the approach and execution, list limitations of the study that affected its outcomes and suggest the future work to be done on the subject.

The thesis used Design Science Research approach to provide both an implementation of a Design System within a software startup and add to the knowledge base on the subject. It allowed me to undergo the actual process while conducting a more general qualitative assessment, which provided a more thorough understanding of the subject overall. In general, the practice worked well, as I managed to provide a useful contribution within the industry along with the recommendations for other software startups.

I also managed to fulfill the research objectives and provide knowledge about the current state of software startups and a starting point for further research. In regard to the suitability of Design Systems for startups, current outcomes showcase that they do have potential to have an impact on work efficiency of the software startups, however, the approach should be adapted to the startup needs and the general nature of a software startup to make it cost efficient and useful.

In order to make the results generalisable to software startups that fit the criteria defined in Chapter 1, I gathered startups with various projects, ages and team sizes for the interviews. The startups were also geographically diverse: one originated from Silicon Valley, one from Norway, one from Slovenia and two from Sweden. During the implementation I compared the most commonly used technologies to make any recommendations wholesome and with merit.

Overall, I believe that both rigor and relevance that form Design Science Research have been demonstrated throughout this study. However, as always there are some

limitations of this research that need to be acknowledged.

6.1 Limitations of the study

Design Systems cover a very broad subject that had to be limited to the scope of this research. Consequently, there is a possibility that the scope itself affected the outcomes by limiting the direction of the research. Part of the limitation is also the time scope of the research, which was limited to the span of 6 months. This provided an obstacle to the evaluation aspect of the implemented Design System. As the Design Systems mainly provide benefits over time, it was not possible to both implement and observe its impact within the time frame. The evaluation is proposed below as future work to be done on the subject.

Another limitation was the amount of the interview participants. Altogether, there were 13 interviewees from 5 startups involved in this part of the research. In retrospect, a potential improvement for a study of this scope would be to try to engage people from more different startups instead of multiple representatives per software startup. That would enable a more effective overview and even more generalisable results. However, this would be a trade-off either way, as that way some aspects of the individual software startup might have gone overlooked.

The interviews were also potentially limited by the available software startups. Since the startups were mainly sampled from the network of the researcher they might share specific characteristics that limit the generalisability of the insights provided. For example, there is a lack of information if the average age, team size or distribution match the averages found in larger surveys. However, the chosen startups do adhere to the predefined criteria range and are positioned across it.

There is also a limitation in the timing of the research and the fast paced technological advancements. As of the time of writing, there are new technologies emerging that have the potential to transform the design and development processes. Those technologies were not the focus of this research due to the lack of their availability and widespread adoption within the industry. One of such technologies is a **design to code automation**, whose aim is to enable direct translation of designs into code. One of the first such attempts for React components is Framer, which was made freely available in May 2020. Therefore, I suggest that those technologies be a subject of further research.

The timing also affected the implementation, because there were some issues with

the used technologies and tools. Those issues affected the efficiency of the implementation and required workarounds, which is not the best practice when building such systems. The best example of such problems is the Storybook version for Web Components that is being released and documented at the time of writing. It could have made an implementation of the single source of truth and documentation of the Design System more thorough. However, I still managed to implement all of the elements as planned, so this was no large limitation.

6.2 Future work

As the Design Systems represent an under researched subject, there is a lot of research that can be done, especially focusing on their usage within software startups. This thesis provides a starting point and sheds light on some interesting directions that would benefit the industry along with the research community.

First, a very important aspect would be doing a long term evaluation of the Design System implementation within a software startups. Such study could provide conclusive insights on the potential of this approach for startups. It could be executed through a field study where the use of the artifact would be monitored in multiple projects and within different startups.

There is also more research needed on the ways to measure the effectiveness of such abstract concepts. As of now, there is very few approaches available that are the result of the individual efforts and not verifiable studies. Such measurement systems could also generally aid the progression of research in this area by providing clear metrics and potentially enabling a more quantitative evaluations.

As mentioned in the limitations, there are also new technologies emerging that enable automated translation of the design to code. They require more research to determine their potential and the impact they will have on the currently widespread practices in software startups and also overall in the industry. Since such technologies are only emerging, research on how other concepts like Artificial Intelligence might be utilised to advance their progress is also needed.

More focused research topics could also go beyond the basic subject of suitability of Design Systems to compare what kind of startups might need such approach the most. Finally, there is a need for broader research to expand the knowledge base on the Design Systems concept itself. An ongoing need is also to standardise it and compare different possible approaches and structures.

Chapter 7

Conclusions

This study was conducted to determine whether the Design Systems are suitable for the software startups and if they have the potential to improve their work efficiency. At the same time, I aimed to open up this field for much needed research on Design Systems. While I conclude that the question of Design Systems suitability cannot be answered universally for all of the software startups out there, they have the potential to solve some reoccurring issues and aid the software startups in achieving their purpose. To determine how fitting they are for an individual software startup should be determined based on its **team structure**, **stage** and most importantly, **the issues** it is struggling with.

The implementation of the artifact within a software startup showed that **it is possible to get a Design System started in moderate time without a large investment of resources**. Even though the long term evaluation was not conducted, over the short term the Design System has proven useful for **more efficient, agile development**. Contrary to the lack of conclusive resources available, the executed interviews have shown that Design Systems are a very relevant topic among the software startups. The awareness and amount of experiences among software startup employees exceeded expectations, especially that they are already using Design Systems in practice.

Overall, the combination of the two methods really contributed to a wholesome outcome. The insights addressed a gap in the knowledge base and formed a good base for further research. Hopefully, this work will inspire more research into Design Systems and help fuel the progress in this fast changing field. The main question raised by this research is, how would a Design System prove itself in a software startup over a longer period of time.

To conclude, there is much yet to be discovered about the Design Systems, espe-

cially in regard to software startups. If anything they are an innovative concept, worth at least a consideration if not direct usage.

"Here's the simple truth: you can't innovate on products without first innovating the way you build them." (Schleifer 2016)

References

Ackoff, R. L. (1962), 'Scientific method: Optimizing applied research decisions'.

Alexander, C. (1977), *A pattern language: towns, buildings, construction*, Oxford university press.

Allan, M. (2015), '20 second gut tests'. Accessed 12.4.2020.
URL: *https://clearleft.com/posts/20-second-gut-tests*

Aveledo, M., De la Rosa, A. & Moreno, A. M. (2010), Usability design recommendations: a first advance, *in* 'Innovations and Advances in Computer Sciences and Engineering', Springer, pp. 395–400.

Beck, C. (2017), 'On the current state of design systems in ux'. Accessed 6.3.2020.
URL: *https://medium.com/better-product/on-the-current-state-of-design-systems-in-ux-4cd0aa1fad71*

Benbasat, I., Goldstein, D. K. & Mead, M. (1987), 'The case research strategy in studies of information systems', *MIS quarterly* pp. 369–386.

Bidelman, E. (2019), 'Shadow dom v1: Self-contained web components'. Accessed 23.5.2020.
URL: *https://developers.google.com/web/fundamentals/web-components/shadowdom*

Bostian, E. (2019*a*), 'Design systems'. Accessed 18.3.2020.
URL: *https://medium.com/swlh/design-systems-b07e5ec6e310*

Bostian, E. (2019*b*), 'Design systems: Design language'. Accessed 20.3.2020.
URL: *https://medium.com/swlh/design-systems-dd8c3b71790b*

Cassidy, T. (2011), 'The mood board process modeled and understood as a qualitative design research tool', *Fashion Practice* **3**(2), 225–251.

Cole, R., Purao, S., Rossi, M. & Sein, M. (2005), 'Being proactive: where action research meets design research', *ICIS 2005 proceedings* p. 27.

Connolly, E. (n.d.), 'The full stack design system'. Accessed 6.3.2020.
URL: *https://www.intercom.com/blog/the-full-stack-design-system/*

Curtis, N. (2010), *Modular web design: creating reusable components for user experience design and documentation*, New Riders.

Curtis, N. (2017*a*), 'Defining design systems'. Accessed 19.3.2020.
URL: *https://medium.com/eightshapes-llc/defining-design-systems-6dd4b03e0ff6*

Curtis, N. (2017*b*), 'Ux patterns $\neq$ ui components'. Accessed 18.3.2020.
URL: *https://medium.com/eightshapes-llc/patterns-components-2ce778cbe4e8*

Darvishy, A. (2014), Accessibility of mobile platforms, *in* 'International Conference of Design, User Experience, and Usability', Springer, pp. 133–140.

Dresch, A., Lacerda, D. & Júnior, J. (2015), 'Design science research: Research method for advancement of science and technology'.

Fanguy, W. (2019), 'A comprehensive guide to design systems'. Accessed 19.3.2020.
URL: *https://www.invisionapp.com/inside-design/guide-to-design-systems/*

Farino, P. (2017), 'Design systems: Zero to one'. Accessed 18.3.2020.
URL: *https://www.youtube.com/watch?v=Eq0-Sz5S9iI*

Frost, B. (2016), *Atomic design*, Brad Frost Pittsburgh.

Ghodeswar, B. M. (2008), 'Building brand identity in competitive markets: a conceptual model', *Journal of product & brand management* .

Gibbons, S. (2016), 'Design thinking 101'. Accessed 18.3.2020.
URL: *https://www.nngroup.com/articles/design-thinking/*

Gould, J. D. & Lewis, C. (1985), 'Designing for usability: key principles and what designers think', *Communications of the ACM* **28**(3), 300–311.

Gubrium, J. F. & Holstein, J. A. (2001), *Handbook of interview research: Context and method*, Sage Publications.

Hacq, A. (2018), 'Everything you need to know about design systems'. Accessed 27.2.2020.
URL: *https:// uxdesign.cc/ everything-you-need-to-know-about-design-systems-54b109851969*

Hallock, J. (2003), 'Colour assignment', *Recuperado el* **10**.

Hammarberg, K., Kirkman, M. & de Lacey, S. (2016), 'Qualitative research methods: when to use them and how to judge them', *Human Reproduction* **31**(3), 498–501.
URL: *https:// doi.org/ 10.1093/ humrep/ dev334*

Hart, G. (2000), 'The style guide is "dead": long live the dynamic style guide', *Intercom* **47**(3), 12–17.

Hassan, C. (2018), 'Ebury chameleon as an example of a design system'. Accessed 20.3.2020.
URL: *http:// labs.ebury.rocks/ 2018/ 03/ 16/ ebury-chameleon-example-design-system/*

Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004), 'Design science in information systems research', *MIS quarterly* pp. 75–105.

Jaeger, R. G. & Halliday, T. R. (1998), 'On confirmatory versus exploratory research', *Herpetologica* pp. S64–S66.

Johanessen, J. (2018), 'Yes, your startup is ready for a design system'. Accessed 21.3.2020.
URL: *https:// uxdesign.cc/ yes-your-startup-is-ready-for-a-design-system-7c50d13e1516*

Kholmatova, A. (2017), *Design Systems: A practical guide to creating design languages for digital products*, Smashing Media.

Kitchenham, B. A., Budgen, D. & Brereton, O. P. (2011), 'Using mapping studies as the basis for further research–a participant-observer case study', *Information and Software Technology* **53**(6), 638–651.

Knapp, D. (1999), *The brand mindset: five essential strategies for building brand advantage throughout your company*, McGraw Hill Professional.

MacDonald, D. (2019), 'Practical ui patterns for design systems'.

Mail, R. (2011), 'Royal mail logo and brand guidelines'. Accessed 20.3.2020.
URL: *https:// identitydesigned.com/ royal-mail/*

March, S. T. & Smith, G. F. (1995), 'Design and natural science research on information technology', *Decision support systems* **15**(4), 251–266.

Mauduit, M. (2019), 'Design system vs startup. takeaways.'. Accessed 17.3.2020.
URL: *https:// medium.muz.li/ design-system-at-buzzvil-ca3eecd971a9*

McGowan, E. (2018), 'What is a startup company, anyway?'. Accessed 2.3.2020.
URL: *https:// www.startups.com/ library/ expert-advice/ what-is-a-startup-company*

Moroni, I., Arruda, A. & Araujo, K. (2015), 'The design and technological innovation: how to understand the growth of startups companies in competitive business environment', *Procedia Manufacturing* **3**, 2199–2204.

Newcomer, K. E., Hatry, H. P. & Wholey, J. S. (2015), *Handbook of practical program evaluation*, Wiley Online Library.

Nielsen, J. (1993), 'Iterative user-interface design', *Computer* **26**(11), 32–41.

Nielsen, J. (2006), 'Corporate ux maturity'. Accessed 17.3.2020.
URL: *https:// www.nngroup.com/ articles/ ux-maturity-stages-1-4/*

Nisen, M. (2014), 'What makes a startup, statistically?'. Accessed 3.3.2020.
URL: *https:// qz.com/ 313532/ what-makes-a-startup-statistically/*

Palmer, T. (2019), '2019 design tools survey'. Accessed 23.5.2020.
URL: *https:// uxtools.co/ survey-2019/*

Park, W., Han, S. H., Kang, S., Park, Y. S. & Chun, J. (2011), 'A factor combination approach to developing style guides for mobile phone user interface', *International Journal of Industrial Ergonomics* **41**(5), 536–545.

Punchev, Y. & Williams, O. (2019), *Getting started with Design Systems*, Marvel Prototyping Ltd.

Pyrhönen, E. (2019), *Hack the Design System*, Idean Publishing.

Rastelli, C. (2019), 'Measuring the impact of a design system'. Accessed 21.3.2020.
URL: *https:// medium.com/ @didoo/ measuring-the-impact-of-a-design-system-7f925af090f7*

Reis, E. (2011), 'The lean startup', *New York: Crown Business* p. 27.

Richards, L. & Morse, J. M. (2012), *Readme first for a user's guide to qualitative methods*, Sage.

Rouse, M. (2013), 'What is skeuomorphism?'. Accessed 9.6.2020.
URL: *https://whatis.techtarget.com/definition/skeuomorphism*

Saakes, D., Yeo, H.-S., Noh, S.-T., Han, G. & Woo, W. (2016), Mirror mirror: An on-body t-shirt design system, *in* 'Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems', pp. 6058–6063.

Saring, J. (2019*a*), '7 tools for building your design system in 2020'. Accessed 9.6.2020.
URL: *https://blog.bitsrc.io/7-tools-for-building-your-design-system-in-2020-452d9c9b3b8e*

Saring, J. (2019*b*), 'Should your startup build a design system?'. Accessed 18.3.2020.
URL: *https://blog.bitsrc.io/should-your-startup-build-a-design-system-4ec719534279*

Schleifer, A. (2016), 'The way we build'. Accessed 10.6.2020.
URL: *https://medium.com/airbnb-design/the-way-we-build-511b713c2c7b*

Seers, K. (2012), 'Qualitative data analysis', *Evidence-based nursing* **15**(1), 2–2.

Sharma, S. & Shirisha, P. (2010), An intelligible representation method for software reusable components, *in* 'Innovations and Advances in Computer Sciences and Engineering', Springer, pp. 221–225.

Shawn Lawton, H. (2018), 'Web content accessibility guidelines (wcag) overview'. Accessed 20.3.2020.
URL: *https://www.w3.org/WAI/standards-guidelines/wcag/*

Shawn Lawton, H. (2019), 'Introduction to web accessibility'. Accessed 20.3.2020.
URL: *https://www.w3.org/WAI/fundamentals/accessibility-intro/*

Shirali-Shahreza, S. & Shirali-Shahreza, M. (2010), Using formal methods in component based software development, *in* 'Innovations and Advances in Computer Sciences and Engineering', Springer, pp. 429–432.

Simon, H. A. (1996), *The sciences of the artificial (3rd ed)*, MIT press.

Spillers, F. (n.d.), 'Introduction to web accessibility'. Accessed 20.3.2020.
URL: *https://www.interaction-design.org/literature/topics/accessibility*

Suarez, M., Anne, J., Sylor-Miller, K., Mounter, D. & Stanfield, R. (2017), *Design systems Handbook*, Online publishing.

Thornhill, A., Saunders, M. & Lewis, P. (2009), *Research methods for business students*, Prentice Hall: London.

Toman, J. (2017), 'Design systems, style guides, pattern libraries.'. Accessed 19.3.2020.
URL: *https://product-unicorn.com/design-systems-style-guides-all-those-libraries-what-the-hell-is-the-difference-4c2741193fdc*

UXPin (2017), 'Creating a design system: The step-by-step guide'. Accessed 21.3.2020.
URL: *https://www.uxpin.com/create-design-system-guide/create-ui-inventory-for-design-system*

Vesselov, S. & Davis, T. (2019), *Building Design Systems*, Apress.

Wallas, P. (2019), 'Why design systems may not always be suited to startups'. Accessed 17.3.2020.
URL: *https://uxdesign.cc/why-design-systems-may-not-always-be-suited-to-startups-94dd8c4243c*

Walsh, R. (2009), *The web startup success guide*, Apress.

Warren, S. (2012), 'Style tiles'. Accessed 20.3.2020.
URL: *http://styletil.es*

Watson, L. (2018), 'Eu directive on the accessibility of public sector websites and mobile applications'. Accessed 25.6.2020.
URL: *https://developer.paciellogroup.com/blog/2018/04/eu-directive-on-the-accessibility-of-public-sector-websites-and-mobile-applications/*

Wilhelm, A. (2014), 'What the hell is a startup anyway?'. Accessed 3.3.2020.
URL: *https://techcrunch.com/2014/12/30/what-the-hell-is-a-startup-anyway/*

Appendix A

Interview Questions

A.1 Background

- Please state you name, age and current position at the startup.
- Can you tell me what is your current level of education?
- How many years of experience do you have in (insert field) and what were your previous roles?
- What is the startup's work about? How old is it?
- How many employees does the startup have and how many of them are designers and how many developers?
- How many projects involving a web page design and or a mobile application are you currently working on (including internal tools)?

A.2 Work

- How would you describe your work responsibilities and tasks?
- How is the work structured within the company? How does the team communicate? **Sub-question**: How do designers usually communicate with developers?
- **Question for the developer**: Which technologies and tools do you use for work?
- **Question for the designer**: Which tools do you use for different areas of design, like user interface design, prototyping and hand-off?

- **Question for the manager**: Who makes the decisions on technologies and tools used? What is the budget designated for it?

A.3 Encountered challenges

- Where do you encounter most challenges? How often do they appear and how do you deal with them? Are there any communication issues and can you describe them?
- What are the main challenges you encounter as a startup? **Sub-question**: Specifically ask about funding, team size, competition and idea pivoting.

A.4 Design Systems

- I will give you a few different terms and want to know if you know what they mean and how would you describe them. **Terms**: style guide, brand identity, component library, visual design language, usability, accessibility.
- What does a term Design System mean to you?
- Have you ever encountered that term before and in what context (heard of it, used it)?
- **If they used it**: Do you use it now or have you used it before? Can you describe it? What do you use it for? How did you make the decisions on how to build it (if you took part)? What was your experience like? All in all, was it a positive or a negative experience?
- **If they heard but did not use it before**: What were the reasons to not use it? Would you like to use it?
- **If they did not hear of it / use it**: How did you usually deal with the design handoff to the development team? Who made the decisions on how things would look and how did you execute the right looks? Did you structure the design or just do it on the go while developing products?

Appendix B

Startup Testimony

wellbe
by L2GO

Impact Testimony about the thesis work of Neža Đukić
8.6. 2020

Over the past six months, Neža has worked on a Design System for our company. She completed the whole chain for the Design System and built our next mobile app platform herself using it. What she has done is very valuable for us and it is exactly in line with our needs in terms of modularity in combination with flexibility. As a startup, we try to build on accessible and affordable tools. The Design System also helps keeping up a high quality of design while still continuously modifying our interfaces and systems all the time, due to our iterative learning process and business model changing a lot at this early stage.

We aim to continue using the Design System so it can evolve with the brand and as our team expands. So far, we have been able to see its effects during this challenging period when our focus shifted and as we were in need of a rapid execution. The modular nature of the new development process showed a lot of promise to increase the development speed. In addition, we plan to use it to unify the user experience across our services and products and increase the brand awareness overall.

Regards,

David Bauman
Co-founder and CTO at L2GO